

UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
DIPARTIMENTO DI INFORMATICA E COMUNICAZIONE



SCUOLA DI DOTTORATO IN INFORMATICA
Settore disciplinare INF/01

Tesi di Dottorato di Ricerca
CICLO XXIII

**AN ARCHITECTURE FOR END-USER
DEVELOPMENT SUPPORTING GLOBAL
COMMUNITIES**

Barbara Rita Barricelli

Relatore: Prof. Ernesto DAMIANI
Correlatore: Prof. Stefano VALTOLINA

Direttore della Scuola di Dottorato: Prof. Ernesto DAMIANI

Anno Accademico 2009/2010

Abstract

Increasingly organizations require their members to act not only as end users but also as developers of their tools, i.e. to create, shape and adapt the software artifacts they use without becoming computer experts (Lieberman et al., 2006; Costabile et al., 2007b). In this way, they move from being mere consumers to active producers of knowledge and developers of software artifacts (Bruns, 2008). This leads to an evolution of the work environment and the organization and force the designers to adapt the software artifacts to meet the needs of the end users and to manage this co-evolution of users and software (Bourguin et al., 2001; Ginige and De Silva, 2009). Moreover, the achievements of social media, Web 2.0 and the advanced information technologies lead to an upward diffusion of global communities, geographically distributed, that collaborate asynchronously on the same design projects. The members of global communities belong to different cultures, therefore cultural boundaries need to be transcended. The mantra “making all voices heard” (Schuler, 2008) has to be evolved into “making all voices heard and understood” to allow the proper participation of end users to knowledge and software artifacts creation, sharing and evolution. To respond to these challenges, the thesis presents a semiotic model for end-user development and a Web architecture (Barricelli et al., 2009c; Barricelli et al., 2010a) that supports 1) an interaction localized (Esselink, 2000) to end user’s culture, domain of activity and digital platform in use, and 2) the collaborative creation and evolution of knowledge and software artifacts (Dittrich et al., 2009). The architecture is Ajax-like, component-based, Web service-based, and underpins re-use and evolution of software.

Acknowledgments

This thesis is dedicated to the memory of Professor Piero Mussio (1941-2010), my mentor and PhD advisor.

I wish to express my gratitude to Prof. Ernesto Damiani who took over for Professor Mussio as advisor of this thesis. I also wish to express my sincere appreciation to Prof. Stefano Valtolina, co-advisor of this thesis, for his observations, suggestions and constant support. Great thanks go to my referees, Prof. Maria Francesca Costabile, Prof. Yvonne Dittrich, and Prof. Athula Ginige, who dedicated their time to revise my work and made very useful suggestions which helped me in improving this thesis. Many thanks to Antonio Piccinno for his valuable suggestions and support. I am indebted to my former colleagues Andrea Marcante, Marco Padula, Loredana Parasiliti Provenza and Paolo Luigi Scala and to my current colleagues Li Zhu and Claudia Iacob for the research done together at Computer Semiotics Laboratory (CSLab) and, with some of them, at Consiglio Nazionale delle Ricerche (ITC-CNR). I am also obligated to Daniela Fogli and Giuseppe Fresta for their work on SSW and BANCO. I also wish to thank all the students and domain experts who participated in the evaluations of the prototypes. I would like to express my heartiest thanks to my mother and my grandmother. I also wish to express my gratitude to my father, Martina and Matteo. My thanks are also due to Enrico and his family. My sincere thanks go to my closest friend Giorgia for sharing the Stockholm Syndrome, the Resistance and the Uprising. Last but not least, I would like to thank with all my heart Federico who always walks along with me and brings love into my life.

Table of contents

Abstract.....	2
Acknowledgments	3
Table of contents.....	4
List of Figures.....	8
Introduction.....	12
Part I: Research context	15
Chapter 1: End-user development.....	16
1.1. Origins of end-user development	16
1.2. End user definitions.....	17
1.3. End-user development definitions.....	21
1.4. End-user development activities.....	23
Chapter 2: Cultures of participation.....	27
2.1. Computer supported cooperative work	27
2.2. Cooperative, participatory and meta design	29
2.3. Collaborative design on the Web	33
2.4. Communication and knowledge sharing in collaborative design.....	34
2.5. Collaborative design and HCI	36
2.6. Methods and tools to support collaborative design	37
Chapter 3: Global communities	42
3.1. Globalization and participation in the Web.....	42

3.2.	Globalization and communication gap	43
3.3.	Internationalization	44
3.4.	GILT methodological model	48
3.4.1.	Globalization and internationalization	50
3.4.2.	Localization and translation	51
3.4.3.	GILT best practices	52
3.4.4.	GILT in practice	59
Chapter 4:	Challenges and thesis contributions	62
4.1.	Challenges	62
4.1.1.	System customization	62
4.1.2.	System evolution	63
4.1.3.	Communication	63
4.2.	Thesis contributions.....	63
Part II:	Case studies	66
Chapter 5:	Case studies	67
5.1.	Factory automation	67
5.2.	Building construction	69
5.3.	Medical collaboration	72
5.4.	Tourism and cultural heritage.....	74
5.5.	Workflow management	76
Part III:	A semiotic model for end-user development	79
Chapter 6:	Semiotic view on digital communication	80
6.1.	Semiotic view on communication process	80
6.2.	Semiotic view on digital communication process	86
6.3.	Phenomena affecting the digital communication process	88
6.3.1.	Communication gap.....	88
6.3.2.	User diversity.....	89
6.3.3.	Technological grain	89
6.3.4.	Co-evolution	90
6.3.5.	Tacit knowledge and implicit information	92
6.4.	Computer semiotics and HCI	93

6.5.	The role of the software system in digital communication	99
Chapter 7: A semiotic model for end-user development		101
7.1.	A semiotic view on end-user development	101
7.2.	The interactive software system as a mediator	103
7.3.	Mediation process and mediation mechanisms	105
7.4.	The Software Shaping Workshop methodology.....	108
Part IV: An architecture for end-user development.....		113
Chapter 8: BANCO architecture		114
8.1.	B-architecture for EUD supporting global communities.....	114
8.2.	B-architecture and system customization	115
8.3.	B-architecture and system evolution	117
8.4.	B-architecture and communication.....	118
8.5.	B-architecture overview	121
8.5.1.	B-architecture client side.....	121
8.5.2.	B-architecture server side	123
8.6.	B-architecture implementation	124
8.6.1.	BANCO engine and specification documents.....	124
8.6.2.	BANCO knowledge archives	128
8.7.	B-architecture at run time	129
Part V: Evaluation of the architecture.....		131
Chapter 9: Implementing the B-architecture.....		132
9.1.	Factory automation case study	133
9.2.	Building construction case study.....	140
9.3.	Medical collaboration case study	146
9.4.	Tourism and cultural heritage case study	152
9.5.	Workflow management case study.....	159
Chapter 10: Usability evaluations for the SCV case study		164
10.1.	Workshop for tourist.....	165
10.1.1.	Heuristic evaluation	165
10.1.2.	User test.....	168
10.1.3.	Semiotic engineering evaluations	174

10.2. Workshop for publisher	177
10.2.1. Heuristic evaluation	177
10.3. Workshop for domain expert.....	179
10.3.1. User test.....	179
Conclusions and future developments	185
References.....	188

List of Figures

Figure 1. Spectrum of software-related activities. Adapted from (Ye and Fischer, 2007)..	20
Figure 2. Unwitting and witting software developers between pure end users and professional software developers. Adapted from (Costabile et al., 2008b).....	21
Figure 3. CSCW time/space matrix.	28
Figure 4. The star life cycle for human-computer interface development. Adapted from (Hartson and Hix, 1989).	30
Figure 5. An evolution of the starlife cycle of Hartson and Hix. Adapted from (Fogli et al., 2005).	31
Figure 6. The inclusion relation among the activities of the GILT methodological model. Adapted from (Minazzi, 2008).	49
Figure 7. Localization and translation activities on content and package of a product. Adapted from (O'Hagan and Ashworth, 2002).	49
Figure 8. ISO-639-1 and ISO3166-1 codes for languages and countries and the correspondent locales.....	50
Figure 9. An example of translation from English language to Italian language.	52
Figure 10. The same text of Figure 9, not only translated from English to Italian, but also localized to Italian culture.....	52
Figure 11. English and Danish alphabets (uppercase and lowercase).	55
Figure 12. Some examples of Japanese Kanji composition (Mitamura and Mitamura, 1997).	55
Figure 13. Basic Hiragana syllables (Mitamura, 1985).	56
Figure 14. Some examples of punctuation and Marks (Savourel, 2001).	56
Figure 15. A sentence in Thai language where words are not separated by spaces.	57
Figure 16. The representations of numbers in various languages (Savourel, 2001).	57
Figure 17. Two versions of Wikipedia: (a) English and (b) Arabic. The organization of the page follows the writing direction of the language (from left to right for English and from right to left for Arabic).....	58
Figure 18. The first method of implementation of the GILT methodological model.....	60
Figure 19. The second method of implementation of the GILT methodological model.	60
Figure 20. The third method of implementation of the GILT methodological model.....	61
Figure 21. A general methodology of design research. Adapted from (Vaishnavi and Kuechler, 2004).	64

Figure 22. The expected scenario of the factory automation case study. The downward arrows represent the exchange of the software environments among the stakeholders, while the upward arrows represent the annotations sent among them.	68
Figure 23. The building construction sector case study context.	70
Figure 24. The medical collaborative scenario in which two physicians collaborate remotely to the same MRI analysis in order to reach a diagnosis. The HCI expert applies the changes on the environments requested by the physicians.	73
Figure 25. The scenario in the SCV project.	75
Figure 26. The workflow designer designs the workflow to be followed by the operators.	78
Figure 27. The communication process described in (Tondl, 1981) and evolved in (Marcante and Mussio, 2006).	82
Figure 28. The Ogden and Richards semiotic triangle.	82
Figure 29. The Ogden and Richards semiotic triangle revised according to de Mauro (1982).	83
Figure 30. The two communicants reach a common understanding if they assign compatible meanings to the same signal. The two red triangles (denoted by 1 and 2) represent the semantization processes of the two communicants.	84
Figure 31. The digital communication process model. The designer of the software system is not present during the process of interaction between the user and the system s/he developed.	85
Figure 32. The semiotic triangle that joins oral, written, and digital signal interpretation. Adapted from (Mussio, 2009; Valtolina, 2010).	86
Figure 33. A model of digital communication process.	87
Figure 34. The co-evolution phenomenon (Costabile et al., 2006a).	91
Figure 35. The map of computer semiotics proposed by Andersen (1990).	94
Figure 36. The physical-empirical and the syntactic levels in the digital communication process.	96
Figure 37. The semantic level in the digital communication process.	96
Figure 38. The pragmatical level in the digital communication process.	97
Figure 39. The social level in the digital communication process.	98
Figure 40. The metacommunication process between the designer and the end-user developer is illustrated in the top of the picture. The metacommunication between the end-user developer and the end user is depicted in the left part of the picture.	102
Figure 41. Another scenario of metacommunication where the end user developer develops a system for her/himself and becomes therefore designer and user of the new system. ...	103
Figure 42. The first two steps of a generic mediation process between an end user and an end-user developer.	105
Figure 43. The SSW network. The arrows describe the communication processes that take place among the various stakeholders working at the different levels (Costabile et al., 2007b).	110
Figure 44. An overview of BANCO architecture.	122
Figure 45. The schema for the IM ² L language.	125
Figure 46. An example of “operator” definition given in document D1.	126
Figure 47. An example of localization definition for an “operator” given in document D2.	127
Figure 48. The example of D3 document for the “operator” entity.	127

Figure 49. An example of RDF document for a thread of notes that constitutes an annotation. Two distinct notes are present and are created by two different authors in two different times but referring to the same point of interest on the map of Valchiavenna region.	128
Figure 50. The SSW network for the factory automation case study.	133
Figure 51. An application developed for a factory automation company. The letters have been added on the screenshot for the sake of explanation in the text (Barricelli et al., 2009c).	135
Figure 52. Unwitting programming: the shop foreman drags and drops the entity “bottoniera di sistema” (system button panel) into the canvas representing the background of the assembly-line operator workshop being created.	136
Figure 53. The shop foreman is creating the assembly-line operator workshop shown in Figure 51. The button “automatico” (automatic) is located on the operative button panel.	137
Figure 54. The assembly-line operator annotates the assembly line workshop to communicate her/his problems to the designers. The circle has been added on the screenshot for the sake of explanation in the text.	138
Figure 55. The BANCO architecture implementation for the factory automation case study.	139
Figure 56. The SSW network for the building construction case study.	140
Figure 57. The workshop for the foreman presents her/him the documents (room maps) stored in the archive.	142
Figure 58. After the selection of a map performed by the foreman, the workshop displays the selected map (“STANZA 1”).	143
Figure 59. A visual link is materialized on the map to signal the presence of an annotation.	144
Figure 60. The workshop for the operator in the technical office.	145
Figure 61. The BANCO architecture implementation for the building construction case study.	146
Figure 62. The SSW network for the medical collaboration case study.	147
Figure 63. System workshop W-ReprNeuRa used by a neuroradiologist who creates the application workshop devoted to the other neuroradiologists at use level.	148
Figure 64. System workshop W-CompNeuRa used by a neuroradiologist who is in charge of creating components for the workshop devoted to the neuroradiologists who operate at use level.	149
Figure 65. The workshop for an Italian neurologist. The interface is localized to her/his culture, role and digital platform in use.	150
Figure 66. The workshop for a Hebrew neurologist. The interface is localized to her/his culture, role and digital platform in use.	151
Figure 67. The BANCO architecture implemented for the medical collaboration case study.	152
Figure 68. The SSW network for the tourism case study.	153
Figure 69. The workshop used by the publisher in order to associate to the maps the contents created by the domain experts. The publisher is also in charge of moderate the annotations created by the tourists.	154

Figure 70. The workshop used by the domain expert in order to develop components for the workshop for the tourists. In particular, in this picture the domain expert is building a workbench by selecting operators from the available repositories.....	155
Figure 71. An Italian tourist accesses the Valchiavenna map using her workshop and creates an annotation using an emoticon that expresses appreciation respect to the annotated point of interested.....	155
Figure 72. A Japanese male tourist accesses the Valchiavenna map using his workshop and creates an annotation using an emoticon that expresses appreciation respect to the annotated point of interested.....	156
Figure 73. The scale of liking used in the SCV case study. Four different emotions are materialized with different shapes and colors according to Italian and Japanese cultures.	157
Figure 74. The BANCO architecture implementation for the tourism case study.	158
Figure 75. The SSW network for the workflow management case study.	159
Figure 76. The Workflow Designer-TMS Editor system for semi-automatic workflow composition.....	161
Figure 77. The BANCO architecture implementation for the workflow management case study.....	163
Figure 78. The problems detected in the heuristic usability evaluation performed on the workshop for tourist, classified according to the class of problems they belong to.	167
Figure 79. The classification of the problems detected in the heuristic usability evaluation according to the class of area of intervention they belong to.	168
Figure 80. The profile of the users involved in the user test.	169
Figure 81. The users' computing knowledge and Web use information emerging from the initial questionnaire.....	170
Figure 82. The tasks execution times and averages of each user in the three groups.....	173
Figure 83. The tasks execution times and averages of each group.....	174
Figure 84. The frequency of tags and patterns in the CEM evaluation of the workshop for tourists.....	177
Figure 85. The problems detected in the heuristic usability evaluation performed on the workshop for publisher, classified according to the class of problems they belong to.	178
Figure 86. The classification of the usability problems of the workshop for publisher according to the class of area of intervention.	179
Figure 87. Gender, age, and education level for the group of users involved in the user test for the workshop for the domain expert.....	180
Figure 88. Programming and macro skills, and programming languages used by the domain experts involved in the user test.....	181
Figure 89. Data about the use of the computer by the domain expert. Time spent on the PC, on the Web and social interactive technologies used by them.	182
Figure 90. The tasks execution times and averages of each user involved in the user test.	183

Introduction

Increasingly organizations require their members to act not only as end users but also as developers of their tools, i.e. to create, shape and adapt the software artifacts they use without becoming computer experts (Lieberman et al., 2006; Costabile et al., 2007b). In this way, they move from being mere consumers to active producers of knowledge and developers of software artifacts (Bruns, 2008). This leads to an evolution of the work environment and the organization and force the designers to adapt the software artifacts to meet the needs of the end users and to manage this co-evolution of users and software (Bourguin et al., 2001; Ginige and De Silva, 2009). Moreover, the achievements of social media, Web 2.0 and the advanced information technologies lead to an upward diffusion of global communities, geographically distributed, that collaborate asynchronously on the same design projects. The members of global communities belong to different cultures, therefore cultural boundaries need to be transcended. The mantra “making all voices heard” (Schuler, 2008) has to be evolved into “making all voices heard and understood” to allow the proper participation of end users to knowledge and software artifacts creation, sharing and evolution. To respond to these challenges, the thesis presents a semiotic model for end-user development and a Web architecture (Barricelli et al., 2009c; Barricelli et al., 2010a) that supports 1) an interaction localized (Esselink, 2000) to end user’s culture, domain of activity and digital platform in use, and 2) the collaborative creation and evolution of knowledge and software artifacts (Dittrich et al., 2009). The architecture is Ajax-like, component-based, Web service-based, and underpins re-use and evolution of software.

The approach adopted for the definition of the semiotic model for EUD and for the implementation of the architecture stems from the critical analysis of the existing literature and previous EUD experiences developed in the Computer Semiotics Laboratory (CSLab) in which the thesis was developed, and from the experiences carried out during the thesis work. The approach is collaborative and evolutionary in that the evaluation of each achievement becomes the central activity. This approach respects the rules of Web 2.0, in fact software artifacts 1) are developed as incremental prototypes, in that at each step of development new features are added to them (component-based development), 2) are considered perpetual beta, in that they are no more considered as products but as processes of engagement that involve also the end users, 3) are developed permitting to open data and services for being reused by others (Web service-based development), and 4) reside in space between devices, i.e. they are not residing only on clients or on servers (Ajax-like). Internationalization and localization techniques are used in the model, to respond to the needs of EUD and global communities requirements. End-user knowledge creation, evolution and management is achieved through the use of pro-active annotation tools, which support the organization of the knowledge.

The thesis is developed in five parts and a conclusion section.

In the first part, the research context is described. End-user development, cultures of participation and global communities concepts are presented through a literature review. As conclusion of this first part, the challenges arising from this research context are highlighted and the thesis contributions briefly introduced.

In the second part, the case studies faced in the last years are presented. They regard different application domains, e.g. medical collaboration, tourism and cultural heritage.

The third part presents a semiotic view on digital communication process and defines a semiotic model for end-user development. The model is communication based, in that it supports the collaborative design and development activity performed by end users, by providing the digital artifacts with communication tools.

The fourth part is dedicated to the Web architecture presentation. The components and mechanisms on which the architecture is build upon and each of its features are presented. Next, the implementation process followed in implementing the architecture is described.

The fifth and last part, describes from the point of view of the implementation, the case studies illustrated in the second part of the thesis. The presentation of the different implementations aims at demonstrating the feasibility and flexibility of the architecture proposed in the thesis. In particular, a case study in tourism and cultural heritage context is used to evaluate the architecture: to this end, the results of several usability evaluations are presented.

Finally, conclusions and future developments are presented.

Part I:
Research context

Chapter 1:

End-user development

This chapter presents an overview on end-user development. Through a literature review, the various definitions of end user and end-user development concepts are presented.

1.1. Origins of end-user development

The so-called *personal computing* was defined by Lehman (1985) as “the use of computing resources by an individual to carry out his or her job”. After a year, Leitheiser and Wetherbe (1986) defined the end-user computing (EUC) as “the use and/or development of information systems by the principal users of the systems’ outputs or by their staffs”, paving the way for the birth of end-user programming (EUP). In fact, as complement for EUC, in the late 1980s EUP approach emerged and evolved. Cypher (1993) defined EUP as an approach in which end users write computer programs without having the need to use conventional programming language and cited, as examples, spreadsheet users who write formulas and macros. Since end users noticed that software developed by professional developers often did not meet their needs and needed to be improved, professional developers began to create software applications flexible and flexible in order to meet end users’ needs. The flexibility of the software applications led to what is called end-user tailoring (EUT), defined by Henderson and Kyng (1991) as “the approach that allows users to modify the artifact they use by changing their stable aspects”. Trigg et al. stated in (1987) that a system could be defined “flexible” if it provides generic objects and

behaviors that can be interpreted and used in different ways by different users for different tasks, while a system could be defined “tailorable” if it allows users to change the system itself for example by building accelerators, specializing behavior or adding functionalities.

Therefore, EUP, EUC, and EUT can be seen as subsequent attempts to realize what is today called end-user development (EUD). As described by Spahn et al. (2008), the term EUD has evolved over time and complements many of other research fields; they consider in fact end-user computing (EUC), end-user tailoring (EUT), and end-user programming (EUP) as predecessors of EUD. In Ko et al. (2010), a review of end-user programming and end-user development definitions is given and their relations with end-user software engineering (EUSE) are illustrated. The authors define EUSE as EUP involving also activities that address software quality issues (e.g. reliability, efficiency, usability). Moreover, they claim that EUD has the same basic meaning as EUSE but also implies user participation in the software development process.

EUD activities allow end users, that often feel frustrated because of the difficulties they encounter interacting with the system they use in order to perform their activities, to create or modify software artifacts, so to obtain a better support for their work (Costabile et al., 2006b).

1.2. End user definitions

Cypher (1993) defined the end user as “a user of an application program”. Typically, the term means that the person is not a computer programmer. A person who uses a computer as part of daily life or daily work, but is not interested in computers per se”, and by Nardi (1993) as “the person who does not want to turn a task into a programming problem, who would rather follow a lengthy but well-known set of procedures to get the job done”.

End users are “people outside the information system department”, required “to develop software applications in support of organizational tasks” (Brancheau and Brown, 1993). With the term “end users”, the users of computer tools are confined at the “end of the process of computer programming, far removed from the programmer” (Cypher, 1993).

These people use computer systems as part of daily life or daily work, but are not interested in computers per se (Costabile et al., 2007b). End-user development techniques propose various approaches that allow “users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend software artifacts” (Lieberman et al., 2006). In this situation, end users are increasingly evolving from passive consumers of data and computer tools into active producers of information and software (Costabile et al., 2007a; Fischer, 2002).

End users are not necessarily experts in computer science, but in the domains they work and act (e.g. industrial organizations, business organizations), they are responsible for possible errors and mistakes, even those generated by wrong or inappropriate use of the software they develop. Also when given this responsibility, they do not willingly become computer experts, although they need to program and maintain control over the information-processing tasks they generate (Costabile et al., 2006b). To overcome this contradiction, several researchers proposed approaches in which end users can program following their reasoning habits, and not computing habits of computer scientists, embedded in traditional programming languages (Mussio et al., 1991; Reppenning and Ioannidou, 2006; Whitley and Blackwell, 2001).

Ásand and Mørch consider end users those persons who are skilled with computers. They gave in (2006) a classification of the actors involved in companies in which EUD activities are performed. The first category of users that Ásand and Mørch recognized is the one of “regular users”, i.e. workers who want to use the tools offered by the system they use in order to perform their tasks, but who are not interested in tailoring the system itself. Another category is the “super users” one, constituted by domain-trained workers, skilled with computers, who are available to teach other users how to use the system and that are interested in exploring meta-tools in the free time. “Local developers” are instead domain-trained workers, more skilled than the “super users” and with programming knowledge. And finally, the last category is the “Professional developer” one, in which IT workers develop new software applications or new versions of existing ones. Also Nardi and Miller (1990) gave a definition of users’ category, by distinguish them considering their skill level and grouping them into three groups: 1) non-programmers, who are not formed in

programming or who have just a little programming knowledge, 2) local developers, as workers that are familiar with some applications and act as consultant for non-programmers, and 3) programmers, who know at least one programming language, are skilled in computing and act as consultant for local developers.

At some point, it is possible to recognize some similarities between the two classifications, the one of Ásand and Mørch and the other of Nardi and Miller. In fact, the Ásand and Mørch's "Super user" category together with the "Local developers" one could be assimilated to the Nardi and Miller's "Local developers" one, and furthermore, another similarity could be identified between "Professional developers" as described by Ásand and Mørch and "Programmers" seen by Nardi and Miller's point of view. But there is a difference between the definitions of "Regular users" and "Non-programmers": Ásand and Mørch, in fact, describe regular users as those workers who are not developers and not interested in tailoring but also in using their systems to perform they daily work, while Nardi and Miller define the non-programmer users as workers who could also some have programming skills.

A more extended classification of end users, in which many other differences are evident, is the one given by Rockart and Flannery (1983). They grouped them into six main categories: a) "Non-programming end users" who access data stored in computers using limited software applications provided by others. They do not program and do not use report generators either; b) "Command level users" who access data by performing simple queries and by generating reports. They can access and manipulate information stored into databases by using limited set of commands from languages such as FOCUS, SQL, or SAS; c) "End-user programmers" who program both with command and procedural languages to develop their own applications; d) "Functional/support personnel" who are programmers that support other end users; e) "End-user computing support personnel" who are fluent in end-user languages and, in addition to aiding end-users, also develop either application or "support" software; f) "DP programmers" who program in end-user languages. Rockart and Flannery, unlike Ásand and Mørch and Nardi and Miller, consider all the actors involved into the companies' dynamics as end users, and ascribe to all of them, except for the "non-programming end users", some programming skills but they

describe no one of them as professional developer, because also the higher category, the “DP programmers”, considers only workers who program in end-user languages. Also the problem of multiculturalism of workspaces is one of the factors that contribute to users’ categories definitions: as illustrated by Bødker and Pedersen (1991) the problem of multicultural workspaces stating that in a system the culture is not explicit but implicit because it is “hidden behind or in the various artifacts, symbols, work routines, and established patterns of cooperation”. Furthermore, as observed by Costabile et al. (2003), several categories of end users may be recognized also taking into account their abilities and disabilities. The “User diversity” is in fact one of the phenomena that affect the HCI process and described in (Costabile et al., 2006b): users do not belong to a uniform population but constitute communities that are characterized by different cultures, goals, and tasks, and each community develops particular abilities, knowledge, languages, and notations. The authors stated therefore that during system design the user diversity phenomenon has to be taken into account in order to avoid that some users may be forced to adopt specific languages related with the domain, but different from their own, making the interaction process difficult. On the contrary, Ye and Fischer (2007) tried not to classify users into categories, but to observe how the distinction between users and developers is going to disappear: with Figure 1, the authors represented at the left side of the spectrum the end users, who they describe as the owners of the problem, and at the right side the professional software developers, defined as the actors who build software systems of end users.

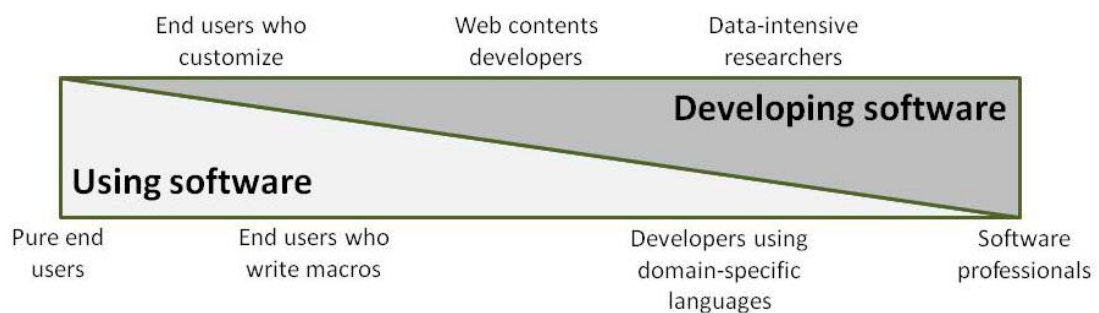


Figure 1. Spectrum of software-related activities. Adapted from (Ye and Fischer, 2007).

The authors put in the middle of the spectrum those actors “who have certain software development skills but that are not interested in software per se” because they don’t develop software for other people, they only develop software in order to solve specific needs that they own. They call these people “domain experts” after the definition given by Costabile et al. (2006b) that stated that those people are experts in a specific domain but that they are not necessarily experts in computer science and that they use software environments to perform their daily tasks.

Costabile et al. extend Ye and Fischer’s spectrum in (Costabile et al., 2008b) by introducing the *unwitting software developers* (Figure 2) as actors in the spectrum of activities. This concept derives from Petre & Blackwell (2007) research on children seen as end-user programmers, for whom the goal is not programming but playing, constructing and deconstructing.

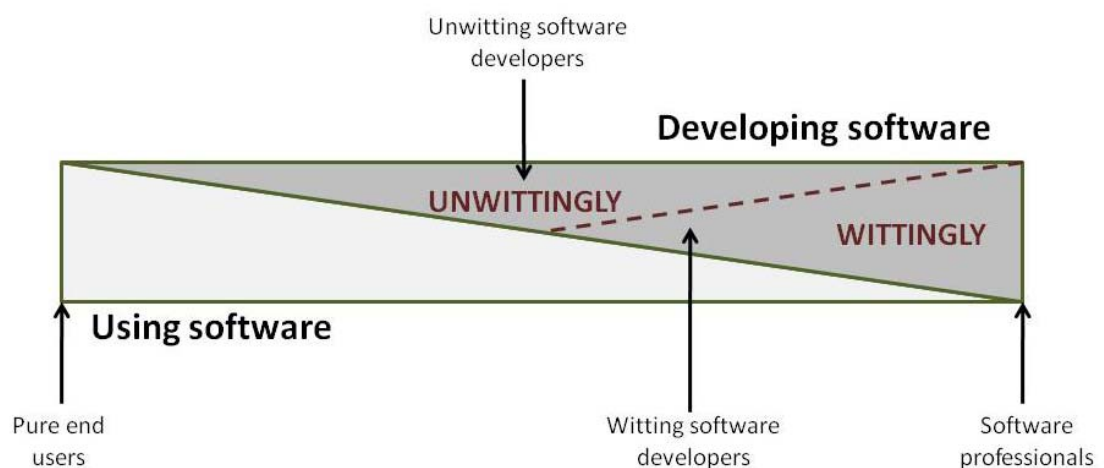


Figure 2. Unwitting and witting software developers between pure end users and professional software developers. Adapted from (Costabile et al., 2008b).

1.3. End-user development definitions

EUD-Net (EUD-Net), a network of Excellence on End-User Development, funded by the European Community, proposed the following definition for end-user development:

End-user development is a set of activities or techniques that allow people, who are non-professional developers, at some point to create or modify a software artifact.

The members of EUD-Net tried to emphasize that the end user is a person who has not development skills but who needs at some point to be a developer and to create tools that could be useful to her/his work or to modify tools that have been created by someone else. In this definition the EUD-Net explicitly refers to some “activities” and “techniques” that enable the end user to develop. In 2006, the research performed by the members of EUD-Net leads to the publication of the book edited by Lieberman et al. (2006). In the introduction of this book, the editors extended the definition of EUD given by the EUD-Net network, defining EUD as

a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact.

Besides the creation and the modification of software artifacts, Lieberman et al. (2006) considered also the extension activity so to enable the end user to complete the software applications developed by someone else by adding new functionalities that her/his expertise suggests as useful. Lieberman et al. did not refer to activities and techniques but introduced the words “methods”, “techniques”, and “tools”, giving to the definition a more concrete direction.

The “old computing” as claimed by Shneiderman (2002) is focused on what computers can do for the user, while the “new computing” regards people activity and what people can do by using computers. Computer users are increasingly evolving from passive consumers of data and computer tools into active producers of information and software (Fischer, 2002; Costabile et al., 2007b). End-user development (EUD) techniques meet this demand by proposing various approaches.

With end-user programming, end-user computing, and end-user tailoring the existing distance between end users and developers has been reduced but the two figures were not

completely recomposed in one: the birth of end-user development represented the first big step toward this rejoin. Andersen and Mørch (2009) consider how the different approaches to EUD vary with respects to how methods, activities, techniques, and tools are emphasized and whether they focus on creation or modification of software artifacts. They also coped with problem of “the why” of EUD, by illustrating it from different points of view: human-computer interaction (HCI), software engineering (SE), and organizational use. From the HCI perspective, EUD “is about leveraging the deployment of easy-to-use ICT and turning them into easy-to-further-develop systems”, from the SE point of view, “EUD is supportive of the trend of producing generic applications”, and from the organizational use position, “the rationale for EUD is associated with the user diversity found in organizations employing advanced ICT”.

1.4. End-user development activities

As Nielsen (1993) wrote, the use of a system changes the users themselves and as they change they will use the system in different and new ways. This is what is called co-evolution of users and systems (Aroni et al., 2002; Bourguin et al., 2001; Carrol and Rosson, 1992; Costabile et al., 2006a; Ginige and de Silva, 2009). Co-evolution stems from users creativity (the users have some needs and they try to satisfy them exploring novel ways to use the system), and from user acquired habits (users insist in following some interaction strategies to which they become accustomed) (Costabile et al., 2006a).

End-user development covers a large area of interests, i.e. customization of applications by parameters setting, control of a complex device like a home-based heating system, script of interactive Web sites (Sutcliffe and Mehandjiev, 2004). EUD allows users to configure, adapt, and evolve their software by themselves (Pipek et al., 2009) and such tailoring activities, together with personalization, extension, and customization are defined in literature in different ways, sometimes referring the same concepts and sometimes referring different ones (Mørch, 1997).

Trigg et al. (1987) define a system as adaptable if it “enables user-customizable behavior”. They also state that a system can be adaptable in four different ways: i) flexible systems

provide generic objects and behaviors that can be interpreted and used in different ways from different users to carry out different tasks; ii) parameterized systems offer many alternative behaviors among whom the users can choose; iii) integratable systems can be interfaced to and integrated with other facilities being part of the environment or connected to remote facilities; iv) tailorable systems allow users to modify the system by building accelerators, specializing behavior, or adding functionalities.

Mørch (1997) defines the tailoring activity as a way to bridge the gap between the objects that compose the interface (simple widgets such as menu items, icons, buttons or composite widgets such as menus, dialog boxes) and the underlying implementation code that defines the functionality (written in a general-purpose programming language). Furthermore, he presents three levels of end-user tailoring: by customization, by integration, and by extension. With customization users can modify the appearance of presentation objects (the ones that compose the interface) or can edit their attribute values by choosing among a set of predefined configurations. Integration allows users to add existing functionalities to an existing application. Extension permits to add new functionalities to an existing application. A further definition of tailoring is given by Henderson and Kyng (1991): if the modifications that are being made on a system are on the subject matter of the tool then there is a use activity, otherwise if the modifications are made on the tool itself that can be called activity tailoring.

A study about the relationship between system evolution, customization, configuration and implementation is given in (Dittrich et al., 2009), where Enterprise Resource Planning (ERP) systems are analyzed. Dittrich et al., claim that in ERP systems the concept of customization is not clearly defined, because the practitioners distinguish between various customization categories on the basis of the complexity and difficulties encountered in interacting with them.

A classification of EUD activities has been introduced by Germonprez et al. (2007). They illustrate five characteristics in the functional design of tailorable technologies by adapting what Baldwin and Clark presented in (2000):

- Splitting by reducing a single module to smaller components;

- Substituting by replacing components or parts of them;
- Augmenting by adding new modules;
- Excluding by deleting modules;
- Porting by adding a component made for another technology.

In (Costabile et al., 2006b) another classification of users' activities is presented: the different activities are classified in two distinct classes that includes respectively those activities that allow users to choose among different behaviors by setting some parameters and those activities that imply some programming for software artifact creation or modification. Furthermore, the authors provide examples of activities belonging to the two different classes, Class 1 and Class 2.

Class 1 groups together activities that support the user in setting parameters in order to choose among various behaviors available in the application. Two examples of activities that belong to this class are parameterization and annotation.

Class 2 is constituted by those activities that allow the user to create or modify a software artifact, by programming in any programming paradigm. To meet the users' need of not becoming developers, programming by demonstration, programming by examples, visual programming, and macro generation are used. Examples of activities that belong to this class are modeling from the data, programming by demonstration, formula languages, incremental programming.

In this thesis, the definition of end-user development given by Lieberman et al. (2006), that is the evolution of the definition coined by EUD-Net, is adopted. End users are seen as domain experts, being characterized by a culture that affects their way of working, and who play a specific role in the work context. The architecture presented in this thesis, is

aimed at allowing end-user development activities by implementing visual interactive systems supporting direct manipulation interaction and visual programming.

Chapter 2:

Cultures of participation

Cultures of participation is a recent concept born to describe a context in which consumerism has been surpassed. However, the participation of users in computer science context is not a recent phenomenon. This chapter introduces some of the movements and research areas in which the figure of the end user evolved from passive consumer to active participant and producer, and discusses some problems arising in a collaborative design context.

2.1. Computer supported cooperative work

The term CSCW (Computer supported cooperative work) was first introduced in 1984 during a workshop organized by Irene Greif of Massachusetts Institute of Technology and David Cashman of Digital Equipment Corporation (Grudin, 1994). In (Carstensen and Schmidt, 1999) CSCW is defined as a research field that

addresses how collaborative activities and their coordination can be supported by means of computer systems.

The applications that support the cooperative work are called “groupware” and are defined in (Ellis et al., 1991) as

computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.

Ellis et al. (1991) pointed out the importance of three main aspects affecting the support of cooperative work: communication, collaboration, and coordination.

Several classifications of groupwares on the basis of space and time, has been given in literature (e.g. Grudin, 1994; Johansen 1988; Baecker, 1995), but particularly the matrix given in (Johansen, 1988), and illustrated in Figure 3, is assumed as the more representative.

	same place	different places
same time	Face to face interactions	Remote interactions
different time	Ongoing tasks	Communication and coordination

Figure 3. CSCW time/space matrix.

In the matrix depicted in Figure 3, four regions can be identified:

- **same place/same time**: synchronous co-located interaction (e.g. shared tables, slides presentations)
- **same place/different time**: asynchronous co-located interaction (e.g. project management, team rooms)
- **different places/same time**: synchronous remote interaction (e.g. instant messaging, video conference)
- **different places/different time**: asynchronous remote interaction (e.g. e-mail, group calendars)

In this thesis, the focus is on asynchronous remote interaction, in which persons belonging to the same team collaborate in different times being geographically dispersed.

2.2. Cooperative, participatory and meta design

In traditional software engineering, the life cycle of interactive systems distinguishes between design time and use time. At design time, system developers create environments and tools, figuring out users' needs and objectives. At use time, end users use the system. Traditional design frameworks are based on the assumption that major design activities end at a certain point, after which the system enters use time. End users are active only at use time. Even when performing user-centered design, which requires that the system is designed by iterating a design-implementation-evaluation cycle, development is carried out by software professionals, while end users use the system and, at most, are involved in prototype evaluation (Costabile, 2001).

The design approach adopted in this thesis stems from a different view, rooted in the Scandinavian tradition of cooperative design. The Scandinavian approach to cooperative design is grounded on application development experiences later turned to academic papers (Kyng, 1991). The rationale is that users are “owners” of the problems in their domain and that software engineers are “owners” of the technology and their active cooperation is necessary to develop software artifacts which are usable and accepted by users' communities (Bødker et al., 1988).

This approach falls in the category that was later known in the United States as Participatory Design, defined in (Schuler and Namioka, 1993) as:

a new approach toward computer systems design in which the people destined to use the system play a critical role in designing it.

Schuler and Namioka (1993), presents also the differences between the participatory design approach and the traditional one:

it rejects the assumption that the goal of computerization is to automate the skills of human workers, instead seeing it as an attempt to give workers better tools for doing their jobs.

Participatory design experiences influenced also the view of the star life cycle of software artifacts (Hartson and Hix, 1989), depicted in Figure 4, and lead to the observation that software artifacts design is an evolutive and never-ending process in that the use of the artifacts by the users suggests new uses of the artifacts and requires their redesign (Bianchi et al., 1999).

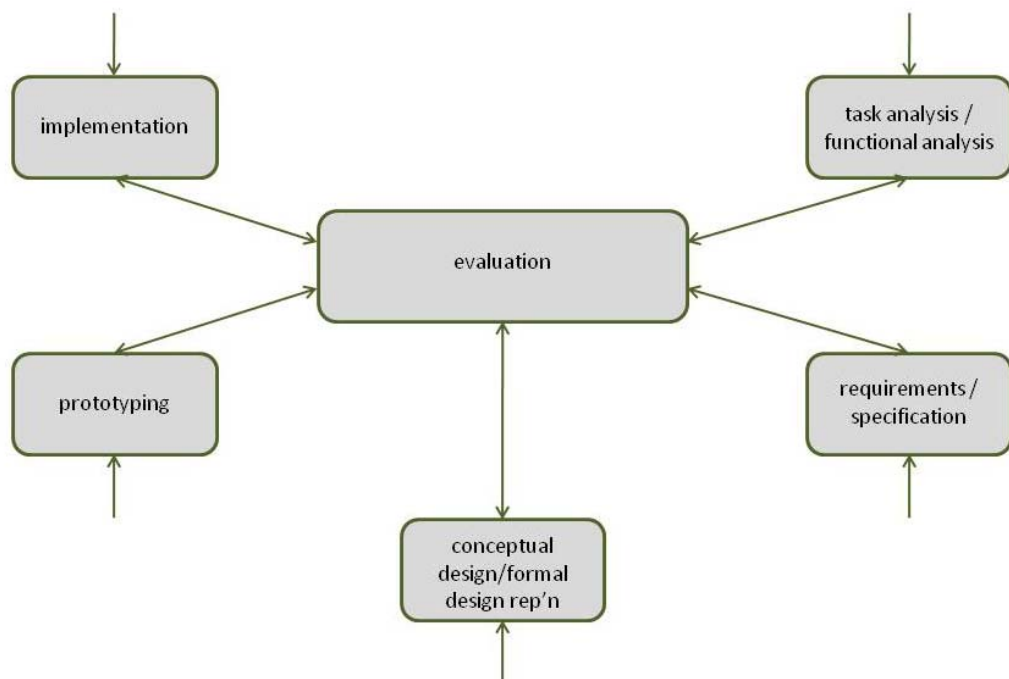


Figure 4. The star life cycle for human-computer interface development. Adapted from (Hartson and Hix, 1989).

An evolution of the Hartson and Hix’s star life cycle, introduced in (Bianchi et al., 1999; Fogli et al., 2005), is presented in Figure 5. In the evolved star life cycle, the “evaluation” activity is detailed considering usability evaluation together with computational and feasibility evaluation. As to the design activity, Bianchi et al. (1999) introduce also the physical design. Furthermore, the authors introduce a new activity, use and maintenance, in order to highlight the importance of the users’ involvement in the design process.

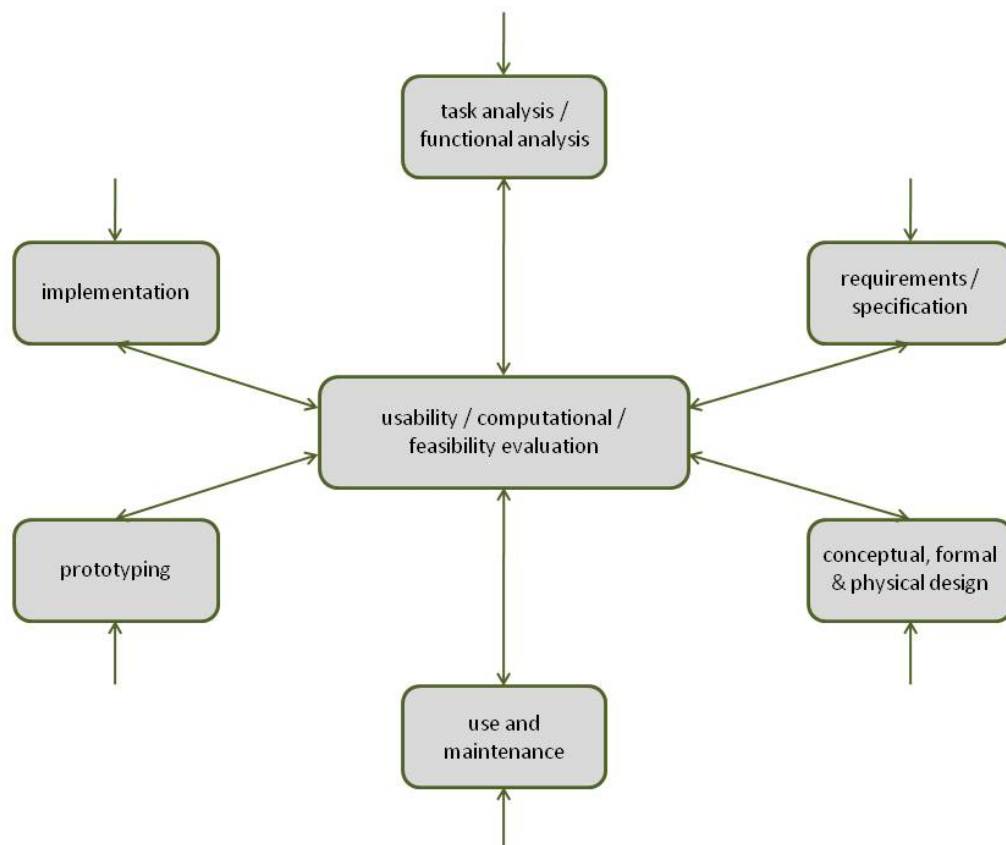


Figure 5. An evolution of the starlife cycle of Hartson and Hix. Adapted from (Fogli et al., 2005).

End-user development puts in practice the end users participation in the overall software life cycle. In this context, activities usually carried out by professional software developers are performed by the end users. End users still use software tools that are provided by professional developers, but they become also co-designers of the tools they use, being actively involved in the continuous development, use and evolution of software artifacts. In order to permit EUD activities, a two-phase process must be considered: in the first phase, software engineers design the design environment. This phase is called meta-design phase and leads to the second phase of the process. In the second phase, end users develop applications for themselves or for other people by using the design environment developed in the first phase. The two phases are not clearly distinct, and are executed several times in an interleaved way, because the design environments evolve both as a consequence of the progressive insights the different stakeholders gain into the design process and as a consequence of the feedbacks provided by end users working with the software artifacts in

the field. This two-phase process requires a shift in the design paradigm, which must move from traditional design and participatory design to meta-design (Costabile et al., 2007b; Bødker et al., 1988).

As defined in (Fischer et al., 2004), meta-design

characterizes objectives, techniques, and processes for creating new media and environments allowing “owners of problems” (that is, end users) to act as designers. A fundamental objective of meta-design is to create socio-technical environments that empower users to engage actively in the continuous development of systems rather than being restricted to the use of existing systems”.

The meta-designers design the design process, which means that they create the social conditions for broad participation in design activities in both design time and use time (Wright et al, 2002).

Therefore, the separation between design time and use time is blurred. These two stages are bridged into a unique “design-in-use” continuum that permits the creation of open and continuously evolvable systems. These systems are extended and/or redesigned at use time by end users collaborating with all the other stakeholders. Therefore, it can be said that the software artifact is in a “perpetual beta” version (Bruns, 2008).

However, in order to involve end users in the development of their tools as part of their own activities, they have to be highly motivated. They have to be actively involved in post-implementation activities by the development team (Wagner and Piccoli, 2007), because their participation represents the main way to solve many problems raised when the final system is used in real work settings, by providing a useful way to fulfill users’ needs and expectations (Costabile et al., 2009).

2.3. Collaborative design on the Web

The Web applications that are nowadays realized, aim at supporting user generated content activities as well as distributed collaborative design. Cultures of participation (Fischer et al., 2004) allow in fact to involve all the stakeholders in the design process to be active producers, system co-designers and supports them to express themselves creatively. They all participate “in personally meaningful activities” (Arias et al., 2000), because it is at use time that conflicts between their needs and the support offered by the existing artifact emerge. Therefore, the artifact must change to adapt to evolving stakeholder needs. The practice of collaborative design and production co-evolve, along with the digital media that enable such practice (Bruns, 2008; Costabile et al., 2006b; Software Engineering Institute, 2010).

Gennari and Reddy (2000) describe the design process as

human activity, involving communication and creative thought amongst groups of participants.

These groups bring into the collaboration different cultures, skills and working habits. To take into account and blend the different contributions, two concepts should be considered: Community of practice and community of interest. Wenger (1998) defines a community of practice (CoP)

a group of people, who work as a community in a certain domain undertaking similar work, sharing practice and addressing common set of problems

such as for example architects, urban planners, software engineers. CoPs develop their own languages and notations to express and communicate their knowledge, ideation, problems and solutions (Carrara et al., 2000). A community of interest (CoI) is defined by Fischer (2001) as a community that

brings together stakeholders from different CoPs bringing their own expertise and background to address design problems of common concern.

CoIs are similar to what Brown and Duguid (2000) define as “communities-of-communities”, whose purpose is to develop their activities learning, sharing and creating knowledge. Examples of CoIs are a community of software designers, HCI specialists, and users interested in the development of musical applications (Borchers, 2001), or a group of geographers and geologists, software engineers and HCI experts interested in water planning problems (Carrara et al., 2000). CoIs are aimed at creating and sharing knowledge to develop collaborative design in a specific domain and are not focused on specific tasks or projects: they differ from project team for their richer purposes, the permanence of composition and duration, which are not linked to any project (Murty, 2010). However, none of CoPs in a CoI has all the knowledge required to achieve these goals, which is tacitly distributed among the various CoPs (Kirsh, 1995).

2.4. Communication and knowledge sharing in collaborative design

In a collaborative design team, stakeholders belong to different CoPs, and each stakeholder owns specific knowledge that is crucial to the design process but not sufficient to solve the whole puzzle. This situation, defined as symmetry of ignorance (Fischer, 2000; Rittel, 1984), requires that each stakeholder’s knowledge is shared and integrated with other stakeholders’ knowledge. Stakeholders in the collaboration act as competent practitioners, in that “they exhibit a kind of knowing in practice, most of which is tacit” and they “reveal a capacity for reflection on their intuitive knowing in the midst of action and sometimes use this capacity to cope with the unique, uncertain, and conflicted situations of practice” (Schön, 1983). Competent practitioners use their (tacit) knowledge to interpret the documents they work from and to understand how to use their tools. For example, a physician performs her/his diagnoses through the application of knowledge and abilities that are personal and developed through the practice.

When using an interactive system, a significant portion of the information conveyed by the system is ‘implicit information’ (Costabile et al., 2006b), i.e. embedded in the actual shape of the elements displayed and in the visual organization of the overall screen image. Such information can only be understood by users who possess tacit domain knowledge, as happens with sequences of images that illustrate sequences of actions to be performed.

The importance of taking into account the different interpretations on what is perceived in the world is highlighted in (Dittrich, 1998; Dittrich, 1997). The author points out how the problems deriving from the communication cannot be solved with a simple translation of concepts from a language to another one. This because our language is part of a practice and it is not separable by it. The meaning of terms and utterances strictly depends on the concrete context of use and cannot be separated from that context.

Sharing and integrating knowledge among different CoPs often fails because similar concepts are expressed with different languages and notations within the various CoPs, while similar expressions are associated with different meanings. In such situations, communication gaps arise between collaborating stakeholders who belong to different CoPs.

Bridging these gaps requires acknowledging the existence of the symmetry of ignorance. It can lead to intelligent and creative results (Engelbart, 1995) but requires the adoption of appropriate progressive semantization techniques (Tondl, 1981) and the creation of adequate software tools.

To meet these needs, Star and Griesemer (1989) introduced the concept of boundary object. A boundary object is an artifact used by different CoPs in their practices but viewed and used differently by each CoP. Boundary objects are used all the time to express ideation and to sustain communication in face-to-face collaboration, for example by using whiteboards and Post-it notes or using wooden models of buildings in architect-client discussions. Blueprints, sketches and drawings are used in design engineering, digital images in medicine and the natural sciences. Papers, electronic documents, and even glossaries play this role in transactions, while sketches are applied to interaction design (Henderson and Kyng, 1991; Verplank, 2009). Stakeholders in the design team interact and

negotiate a concept by using the boundary object as a concrete representation of what they mean. By comparing the different interpretations, acknowledging and discussing the differences determine the emergence of a common experience and shared knowledge (Jennings, 2005). Boundary objects serve as externalizations that capture distinct domains of human knowledge and hold the potential to lead to an increase in socially shared cognition and practice (Resnick et al., 1991). Stakeholders need a space where to carry on these activities: Konkola (2003) introduced the concept of a boundary zone as “an area which resembles a ‘no-man’s land,’ free from prearranged routines or rigid patterns. Andersen and Mørch (2009) define a boundary zone as common ground where CoPs can meet to exchange boundary objects that reflect the outcomes of their collaboration.

2.5. Collaborative design and HCI

In HCI literature the problems deriving from communication gaps between software designers, developers and users are widely documented. The gaps arise since users, designers, and developers might not understand each others’ jargon (Borchers, 2001; Lauesen, 2005; Mahjew, 1992). Interactive systems usually reflect the culture, skills and physical abilities of software engineers rather than those of end users. As a result, end users are obliged to adopt interaction styles that are alien to their culture, and are often charged with housekeeping tasks, which do not interest them and divert their attention from the activity they are performing (Mahjew, 1992; Carrara et al., 2000). Similar gaps arise between HCI experts and software engineers and HCI experts and end users (Costabile et al., 2006b; Folmer et al., 2005). Applications should, quite the contrary be designed to help end users to perform their daily work and activities without being aware of the complex hardware and software technology they are using. Therefore, the interaction style should be adapted to the users’ cultures and capabilities, the context of use, and the performed task (Kuutti, 1996). Several solutions have been proposed to bridge the communication gap and to design usable interactive systems (Borchers, 2001; Costabile et al., 2007b). These solutions explicitly recognize the existence of different cultures and of the communication gap between users and system designers and suggest that the human-system interaction style must be adapted to the end users’ culture and capabilities. To

achieve this goal, they start from the symmetry of ignorance principle and recognize that the collaboration of multidisciplinary experts is necessary to overcome the communication gap (Borchers et al., 2001; Costabile et al., 2006b). Hence, software engineers need to collaborate at the very least, with domain experts, i.e. representatives of the software users. But the collaboration needs to be mediated by HCI experts (Borchers 2001; Carrara et al., 2000; Costabile et al., 2007b; Lauesen, 2005; Folmer et al., 2005). Collaboration between these heterogeneous groups is facilitated by using digital boundary objects. Each digital boundary object is a unique object, augmented by annotation as metadata, which is also a unique event within the message exchanging process. In practice, digital boundary objects are generally used analogously online to how they are used in face-to-face brainstorming, for instance by using multi-touch whiteboards or virtual sticky notes (Branham et al., 2010; Maldonado 2007).

2.6. Methods and tools to support collaborative design

Today, while there have been advances in the tools used, user interface prototyping remains the most effective way to gather requirements, communicate concepts between developers and users and evaluate usability in a cost-effective manner. Rapid prototyping is useful in software engineering to show the developed prototypes to the customers, but professional software tools are required to develop such prototypes, which are not easily operated by end users. The TADEUS project (Stary, 2000) proposes a development methodology starting from a business intelligence model to generate user interfaces or portals by integrating a model-driven, task-based, user-oriented, and object-driven life cycle.

Traditional participatory design approaches exploit techniques derived from social theories (collaborative construction of mock-ups, cooperative prototyping, and game-like design session) that support communication and collaboration within an interdisciplinary team (Bodker et al., 1993). In (Greenbaum and Kyng, 1991), the Future Workshop technique is discussed: it foresees group meetings run by at least two facilitators and having the aim of analyzing common problematic situations, generate visions about the future and discuss how to realize these visions. In this thesis, the term “workshop” is used with the different

meaning provided in (Merriam-Webster online, 2010): “a small establishment where manufacturing or handicrafts are carried on”. Similarly, Cooperative Prototyping is presented (Bødker and Gronbaek, 1991) as an activity where users and designers cooperate. However, prototypes just represent an interactive digital evolution of paper-based mock-ups: real systems are programmed and all modifications require large programming effort that are postponed and made by designers after each session.

Fischer et al. (1994) propose SER (Seeding, Evolutionary growth, Reseeding), a process model to design systems as seeds, with a subsequent evolutionary growth, followed by a reseeded phase; it is adopted to support meta-design and for the development and evolution of the so-called DODEs (Domain Oriented Design Environments), “software systems that support design activities within particular domains and that are built specifically to evolve” (Fischer et al., 1994) which have been evolved toward meta-design in (Fischer et al., 2004). SER is the base of the work described by Carmien et al. (2005), where metadesigners (software engineers) develop a system for caregivers who implement scripts supporting people with cognitive disabilities. Particular attention is paid to the users involved in the domain, and customized environments are provided to them. Co-evolution may also be sustained since tools for automatic feedback and remote observations are used to notify problems to the design team.

The Software Shaping Workshop (SSW) approach (Costabile et al., 2007b) has some similarities with this work, but it emphasizes the need of providing personalized environments to all stakeholders, in terms of language, notation, layout, and interaction possibilities. In SSW, the distinction among professional designers and user is blurred. The focus of SSW is on communicational aspects of HCI and on the communication process among all the stakeholders involved. Moreover the switch is moved from the distinction between design time and use time to the distinction between (meta-) design activities and use time activities. In the SSW approach, the system is organized into various environments (called workshops), each one for a specific sub-community. The design and implementation of application workshops is incremental, in that communities design perpetual-beta artifacts and not final products (Bruns, 2008).

In (Eriksson, 2008) continuous development has been faced, by presenting a study in which end users are involved in design projects. ActionBlocks, a system of ubiquitous and intelligent building blocks is presented. It is a flexible system in that a) end users can change the way in which they use it, b) the interaction designer is able to assemble the system on his own conditions, and c) ActionBlock designer can change or develop the software in a convenient way. The development of such system takes place continuously and the maintenance is part of the development.

Finally, other works focus on experience-centered domains, requiring 6 to 12 years of intensive practice before practitioners achieve the most effective levels of skill (Hayes, 1985). In these domains (e.g. medical diagnosis, chess, professional design, planning tasks, etc.), one of the main challenges in decision support is that users, with different levels of domain experience, have often very different needs; for example, a system designed to satisfy domain experts' specific needs may frustrate novices and vice versa. The SSW methodology emphasizes the need to develop different software environments for end users working in the same domain with different roles.

DAISY (Design Aid for Intelligent Support System), a design methodology for building decision support systems in complex, experience-centered domains (Brodie and Hayes, 2002), provides a technique for identifying the specialized needs of end users within a specific range of domain experience, therefore, it supports the development of customized systems.

DIGBE (Dynamic Interaction Generation for Building Environments) is a system that creates end-user interfaces adapted to the multiple end users with different roles, that collaborate to the management of a building control system (Penner and Steinmetz, 2002). Unlike SSWs, software environments created using DIGBE are adaptive systems that "automatically improve their organization and presentation by learning from visitor access patterns" (Perkowitz and Etzioni, 2000).

Agentsheets (Repenning, 1991) are proposed to introduce an intermediate level of abstraction between high-level building-blocks and conventional programming languages level. They are used to create and extend applications incrementally, not only focusing on

the behavior but also on the look and feel of the artifact. Agentsheets are developed in visual programming domain: they are based on grid structure, their elements are called agents, and each agent's state is represented graphically.

Annotea is a project in the W3C context (Annotea Project) proposing a metadata-based infrastructure, which allows cooperators to annotate Web documents, to share them, to store them in a common repository and retrieve them. Annotea provides protocols for distributed annotation, based on RDF (Resource Description Framework) (RDF), XPointer (XPointer) and HTTP (HTTP), which were adopted in the implementation of Amaya (Amaya), a browser and editor with a plug-in to manage the annotations, and Annozilla (Annozilla), a Mozilla plug-in for annotating Web pages.

Annotations can be visualized in a document context view or in a topic hierarchy view, organized in threads (Koivunen and Swick, 2003). Annotea focuses on the format of annotation and provides a protocol for any annotation performed on a document described according to W3C technologies.

MADCOW (Multimedia Annotation of Digital Content Over the Web) (Bottoni et al., 2004) is an environment developed for annotate any component of a Web page. It provides users with a taxonomy of annotation types to help them to classify every annotation. MADCOW has a client-server architecture: the client is a plug-in for a standard Web browser and the servers manage repositories of annotations which can be accessed by any client.

In Dourish et al. (2000) a new paradigm for document management is proposed based on document properties, rather than on document organization. Properties (including user categorizations, keywords, and links to related items) are the primary uniform means for organizing, grouping, managing, controlling, and retrieving documents; they are meaningful to users and enable to express system activities, such as sharing criteria, replication management and versioning; therefore, they enable the provision of document-based services on a property infrastructure. An experimental prototype, the Placeless documents system has been developed to experiment the proposed paradigm; it is based on three core features: uniform interaction, user-specific properties and active properties. With

the large diffusion of mobile devices such as PDAs, mobile telephones and wireless networks, pervasive systems are becoming the next computing paradigm in which infrastructure and services are seamlessly available anywhere, anytime, and in any format. Pervasive systems are a manifestation of the increasing role played by mobility; Dourish et al. (2007) propose an original view of technology and mobility, focusing diversity and agency as central aspects of a socially-responsible approach to mobile computing. This work also connects current research in HCI, ubiquitous computing and human and social geography suggesting a new foundation for design.

In conclusion, the work of Calvary et al. (2003), which can be classified in the field of context-aware computing, aims at describing a framework for classifying user interfaces supporting multiple targets, or multiple contexts of use. Three facets of the context of use are focused: the end user, the computing platform exploited by the user to carry out his/her interactive tasks and the physical environment where the user is working.

In this thesis, the Software Shaping Workshop approach is adopted. The goal of this research is to support the end user participation and evolution into a role of active producer in collaborative design processes.

Chapter 3:

Global communities

The achievement of globalization process and of new technologies, led to an ever increasing outsourcing phenomenon in organizations. Their members are geographically distributed, and therefore very often do not share the same culture and language. Therefore, it is needed to enable differentiated accesses to the interactive tools they use, in order to allow them to work in a virtual environment suitable to their needs. Such approach is supported by the GILT (globalization, internationalization, localization, translation) methodological model. This chapter first introduces the globalization of cultures of participation and GILT. Theoretical foundations of localization and practical applications are then illustrated.

3.1. Globalization and participation in the Web

The complexity and the expanding scale of most design problems that arise in today's world require more comprehensive knowledge than any single expert can possess. Social creativity and individual creativity need to complement each other, so design becomes a creative, collaborative activity requiring the contribution of a team of several stakeholders. Each stakeholder's knowledge is important for design development. Because no single stakeholder has all the knowledge required, knowledge of the design problem is tacitly distributed among all the stakeholders. They are compelled to communicate and collaborate by sharing this knowledge, but they have different cultural backgrounds, play

different roles in their organizations, and use different notations and languages to express their knowledge. Communication gaps arise, making collaborative design difficult (Snow, 1959). In order to bridge these gaps, stakeholders must agree to teach one another and to recognize: 1) that each member of the team complements others' expertise, 2) that they need to reach a mutual understanding, and 3) that peer collaboration will be indispensable (Rittel, 1984; Arias et al., 2000; Costabile et al., 2006b).

The rise of Web 2.0 opens new possibilities for collaborative design. However, it also creates new barriers to collaborative reasoning and communication. Blogs, podcasts, RSS feeds, social software such as wikis, social networking, and social bookmarking are all geared to creating user-generated content mechanisms, as well as collaborative design environments. Collaborative design on the Web arises from a network of people, most of whom are not computer experts. They may also use different IT platforms, collaborate asynchronously, and be geographically distributed, as they design, create, employ, and evaluate a certain artifact.

In (Fischer, 2009), the emergence of Web 2.0 is described as the event that broke down the boundaries between producers and consumers. Users are in fact attracted to participate in different ways. Some examples of broad-based developments are Wikipedia, YouTube, Second Life but also the open source movement.

3.2. Globalization and communication gap

Collaboration is made difficult by the communication gaps existing among the people involved in a common project. For example, end users and software engineers actually possess distinct types of knowledge, users being the “owners” of the problem and developers being the “owners” of the technology to solve the problem. They follow different approaches and reasoning strategies for modeling, performing and documenting the tasks to be carried out in a given application domain; end users do not understand software developers' jargon and developers often do not understand user jargon. As discussed in the previous chapter, this context in which the knowledge owned by all the

stakeholders involved in a common project is necessary to achieve a common goal, is called symmetry of ignorance (Fischer, 2000; Rittel, 1984).

Clashes among cultures become particularly evident when the system requires end users to perform development activities. The problem is thus how to allow end users to define and develop their applications according to their own style of reasoning and to their mental models of the activities to be performed (Majhew, 1992). Two contrasting requirements arise: (i) the need of guaranteeing an appropriate reasoning environment to every participant to the process; (ii) the need of guaranteeing an appropriate communication among the different participants. As to the first requirement, it is necessary that each participant is allowed to reason, experiment on prototypes and report her/his results using her/his modeling language and notations, which reflect her/his mental models; as to the second, it is necessary that different participants are able to communicate their results to each other and reach an adequate understanding of the reciprocal results (Barricelli et al., 2009b; Barricelli et al., 2009c; Barricelli et al., 2010a).

Moreover, in general end users of a specific interactive system do not belong to a uniform population, but constitute communities, characterized by different cultures, goals, tasks and context of activity (Costabile et al. 2007b). As a consequence, each user community develops peculiar abilities, knowledge and notations and again requires specialized software environments (Mussio et al. 1991; Costabile et al. 2006b). The slogan “one size fits all” does not work; it is well known that people experience many difficulties when they interact with a system designed for a generic user, presenting a huge number of functionalities, being overwhelmed with unnecessary interaction possibilities and often disoriented by them.

3.3. Internationalization

In the context described in the previous section, the World Wide Web plays the fundamental role of medium for international communication, participation, and transaction. The characteristics of the WWW, its tools and technologies support and stimulate the evolution of methods and techniques for interface design for multi-cultured

environments (Barber and Badre, 1998). However, in order to guarantee an international usability, the software applications have to be designed in a culture-oriented way (Reinecke and Bernstein, 2008).

To this end, Yeo (1996) proposes the cultural user interface (CUI) concept. He proposes to create a CUI for each culture that has to be considered when a global software application is designed. To this end, the collaboration of people that belong to the considered cultures is mandatory in order to ensure that the interfaces are acceptable for the target communities. A further extension of CUI is also described, that is the personal user interface (PUI). PUI is based not only on the culture of the users but also on bio-data such as gender, religion, hobbies and educational background that refer to the user personally.

In the global software design literature, many definitions of culture have been given. The one adopted in (Yeo, 1996) is the following:

Culture is defined as behavior typical of a group or class (of people)

Bødker and Pedersen (1991) defined culture as

a system of meaning that underlies routine and behavior in everyday working life

After Borgman (1992), culture is seen as including

race and ethnicity as well as other variables and is manifested in customary behaviors, assumptions and values, patterns of thinking and communicative style

In literature, various cultural models have been proposed and each of them is described by a set of cultural dimensions (Hoft, 1996; Hall, 1959; Trompenaars, 1993; Victor, 1992; Hofstede, 1991).

The most adopted and discussed classification of cultural dimensions is the one proposed by Hofstede (1991). He recognized these five main dimensions:

- 1) Small vs. large power distance: measures the extent to which the less powerful members of organizations and institutions accept and expect that power is distributed unequally.
- 2) Individualism vs. collectivism: measures the degree to which members of organization and institutions are integrated into groups.
- 3) Masculinity vs. femininity: measures the distribution of roles between the genders.
- 4) Weak vs. strong uncertainty avoidance: measures to what extent a culture prepares its members to feel either uncomfortable or comfortable in unstructured situations. Uncertainty-avoiding cultures try to minimize the possibility of unknown and surprising situations by strict laws and rules, safety and security measures, and on the philosophical and religious level by a belief in absolute Truth. Uncertainty-accepting cultures are more tolerant of opinions different from what they are used to; they try to have as few rules as possible, and on the philosophical and religious level they are relativist and allow many currents to flow side by side.
- 5) Long vs. short term orientation: measures to what extent a culture respects values associated with long term orientation or short term orientation. Long term orientation values are thrift and perseverance, while values associated with short term orientation are respect for tradition, fulfilling social obligations.

Several studies have shown how these five cultural dimensions, which classify a person's cultural background into certain scores, relate to certain aspects of a user interface (Dormann and Chisalita, 2002; Marcus, 2001; Smith and Chang, 2003).

Reinecke and Bernstein (2008) proposed an algorithm to calculate these dimensions. User's current and former countries of residence, as well as the duration spent in these countries are considered. The following equation (1) shows how the influence of a country N (IC_N) is calculated by dividing the monthly duration of the user's stay in country N (M_N) by the age in months of the user (A).

$$IC_N = \frac{M_N}{A} \quad (1)$$

Using Hofstede's dimensions (Hofstede, 1991) for each considered country, the score for each user in each dimension is calculated (equation (2)). With the help of Hofstede's five dimensions for each country, the user's score in each dimension can be calculated. The user score for dimension H (UDS_H) is calculated with a sum of all the results of the multiplication of the country score of dimension H (CS_H) with the influence of the country considered (IC_i with i that goes from 1 to N , where N is the total number of countries considered for a user).

$$UDS_H = \sum_{i=1}^N CS_H * IC_i \quad (2)$$

Yeo (1996) categorizes the factors that need to be addressed in the internationalization process of a software into covert and overt. Overt factors are tangible, straight forward and publicly observable elements. Some examples are date, calendars, time, address formats, character sets, punctuation, currency. Covert factors are those elements that are intangible and culture-dependent. Colors, sounds, metaphors are examples of covert factors.

Mahemoff and Johnston (1998), propose a more detailed definition of covert and overt factors. They define overt factors the following elements:

- Time factors (calendar, day turnover)
- Writing issues (character set, text direction, special characters)
- Language issues (word ordering, use of jargon)
- Measures (currency, units of length)
- Formatting (numbers, date and time, number rounding)
- External systems (standard page size)

And covert factors:

- Mental disposition
- Perception
- Social interaction rules
- Context of use

More about the application of internationalization and localization techniques and covert and overt factors follows in the next two sections.

3.4. GILT methodological model

The design and the development process of internationalized products passes through the performance of four distinct activities (Cadieux and Esselink, 2002): globalization, internationalization, localization, and translation. These four activities constitute the so-called GILT methodological model. Figure 6 shows the relationship among the four GILT's activities using inclusion relation. In fact, internationalization, localization and translation are included in the more wide globalization activity in that they are its sub-phases. Internationalization is an activity that is performed independently by localization and translation, because it affects the structure of the product under design and development and not its content. Translation is included in localization because it represents just one of the actions required to localize a product.

The names of the GILT activities are usually referred to with *numeronyms*, i.e. a word in which numbers are used to represent a set of omitted letters. G11N is used for “globalization”, in fact between the first letter “g” and the last letter “n” there are 11 more letters that are omitted in this numeronyms. I18N stands for “internationalization”, L10N is used for “localization” and T9N stands for “translation”.

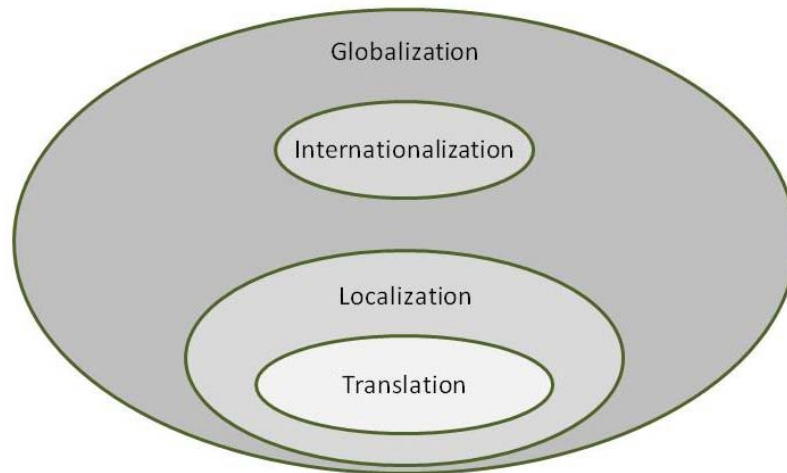


Figure 6. The inclusion relation among the activities of the GILT methodological model. Adapted from (Minazzi, 2008).

The localization activity is detailed in (O'Hagan and Ashworth, 2002), by separating it into two distinct components, content and package. Content is defined as the linguistic structures, while package is the set of all the non-textual elements and the media through which the content is distributed. Figure 7 shows how localization affects both the content and the package of a product, while translation is focused on adaptation of the content only.

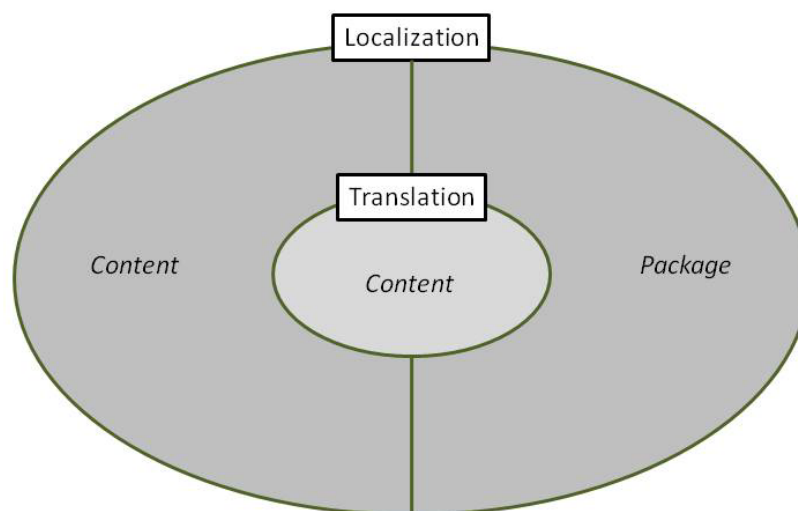


Figure 7. Localization and translation activities on content and package of a product. Adapted from (O'Hagan and Ashworth, 2002).

3.4.1. Globalization and internationalization

Globalization is defined by the Localization Industry Standards Association (LISA, 2010; Esselink 2000) as follows:

Globalization addresses the business issues associated with taking a product global. In the globalization of high-tech products this involves integrating localization throughout a company, after proper internationalization and product design, as well as marketing, sales, and support in the world market.

Globalizing means therefore the extension of a product to different international context making it usable by the different potential users.

The term locale refers to a code that defines language and country for which the product has to be adapted (see Figure 8). It is defined by a language identifier (ISO 639-1) and by a region identifier (ISO 3166-1). The use of these two identifiers allows to distinguish between the same language used in different countries (i.e. UK English and US English).

Language	ISO 639-1	Country	ISO 3166-1	Locale
English	en	United Kingdom	GB	en_GB
		United States	US	en_US
Italian	it	Italy	IT	it_IT
		Switzerland	CH	it_CH
French	fr	France	FR	fr_FR
German	de	Germany	DE	de_DE
Spanish	es	Spain	ES	es_ES
Catalan	ca			ca_ES

Figure 8. ISO-639-1 and ISO3166-1 codes for languages and countries and the correspondent locales.

In (Cadieux and Esselink, 2002), the globalization activity is defined with the equation:

$$G11N = K * L10N \quad (1)$$

Where L10N represents the localization activity, K is the number of locales considered, and the operator * represents the iteration of the localization activity for K times always on the same product but using each time a different locale.

The internationalization activity is defined in (LISA, 2010; Esselink, 2000) as:

the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for re-design. Internationalization takes place at the level of program design and document development.

Equation (1) can be therefore extended as follows:

$$G11N = I18N + (K * L10N) \quad (2)$$

where I18N is the internationalization activity and the operator + stands for the order of the execution of the two activities (internationalization first, followed by localization performed for K different locales).

3.4.2. Localization and translation

Esselink (2000) and LISA (2010) definition of localization is the following:

Localization involves taking a product and making it linguistically and culturally appropriate to the target locale (country/region and language) where it will be used and sold.

This activity regards not only the product itself, but also the entire documentation related to it.

Translation instead is defined in (LISA, 2010; Esselink, 2000) as:

only one of the activities in localization; in addition to translation, a localization project includes many other tasks such as project management, software engineering, testing, and desktop publishing.

The difference between these two activities is well explained with the examples in Figure 9 and Figure 10.



Figure 9. An example of translation from English language to Italian language.

While translation is aimed at maintaining the meaning of original information by exposing them in different languages, localization transforms the information in equivalent ones but adapted to a different culture.



Figure 10. The same text of Figure 9, not only translated from English to Italian, but also localized to Italian culture.

3.4.3. GILT best practices

As suggested by (Madell et al., 1994), in order to develop software suitable for the global market, a two-step process is needed: internationalization of the software first and its localization next.

This process is defined by Yeo (2001) as the global-software development lifecycle (global-SDLC). Global-SDLC supports the maintenance of software in that culture-sensitive elements like dialog messages, error messages and menu names are stored in message files where they are localized. In this way, when a new localization has to be implemented, the software's core does not need to be modified but just a new message file has to be created.

Del Galdo and Nielsen (1996) describe the process of software internationalization as composed by three sequential activities:

- Displaying the native language, character set and notations
- Translating the user interface and documentation so that it is understandable and usable
- Matching the user's cultural characteristics, which goes beyond avoiding offensive icons and must accommodate the way business is conducted and the way people communicate.

In (Russo and Boor, 1993), a cross-cultural checklist that should be considered by interface designers is given. The authors consider several key factors:

- Text: a simple translation is not enough, many aspects should be taken into account. Some of them are:
 - Jargon is confusing and should be avoided
 - Some terms don't exist in all the languages (e.g. computer-related terms)
 - Different languages require different character sets
 - Product names should be translated and adapted to make them fit well in other cultures
 - Number, date and time formats

- Images: images represent the visual language of a culture and therefore not only image recognition but also image acceptability problems should be considered (Cook, 1980; Fussell and Haaland, 1978).
- Symbols: as for images, also symbols have to be acceptable for the target culture.
- Colors: as pointed out in (Thorell and Smith, 1990; Garland, 1982; Courtney, 1986) interpretation of colors varies in the various cultures. Colors play a fundamental role in interface design because they convey information, and therefore they need to be chosen very carefully.
- Flow: the writing system of a language and therefore its reading/writing direction affects the way in which the information is recognized by users on a screen. Hence, the logical flow of what is represented on an interface should follow the proper directions.
- Functionality: sometimes functionalities implemented in software application are not accepted in some cultures because they do not respect the cultural conventions that the user needs.

Savourel extended Russo and Boor's factors and presented in (Savourel, 2001), the aspects that he recognized as fundamental in internationalization process of a product:

- **Character set:** the main ways of representation of texts are alphabet, ideographs and syllabic notation.
 - Alphabets: finite set of characters that once combined form the representation of sounds. In some cases each character may have different forms (e.g. uppercase, lowercase). Figure 11 represents English and Danish alphabets (with both uppercase and lowercase characters).

ENGLISH ALPHABETH
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
DANISH ALPHABETH
ABCDEFGHIJKLMNOPQRSTUVWXYZÆ Ø Å
abcdefghijklmnopqrstuvwxyzæ ø å

Figure 11. English and Danish alphabets (uppercase and lowercase).

- Ideograms: the characters in this kind of writing system represent concepts (Figure 12).



Figure 12. Some examples of Japanese Kanji composition (Mitamura and Mitamura, 1997).

- Syllabic notations: each character represents the sound of a syllable (Figure 13).

1. 46 Basic Hiragana	n/m	wa	ra	ya	ma	ha	na	ta	sa	ka	a	
		ん	わ	ら	や	ま	は	な	た	さ	か	あ
			り		み	ひ	に	ち	し	き	い	
			る	ゆ	む	ふ	ぬ	つ	す	く	う	
			れ		め	へ	ね	て	せ	け	え	
	o	を	ろ	よ	も	ほ	の	と	そ	こ	お	

Figure 13. Basic Hiragana syllables (Mitamura, 1985).

- **Punctuation and marks:** punctuation is applied differently in the various languages. Also different marks exist. Some examples are given in Figure 14.

Script	Symbols
Georgian	paragraph separator ‘∴’
Armenian	exclamation mark ‘՛’, comma ‘՛’, question mark ‘՞’, full stop ‘.’
Arabic	comma ‘٫’, semicolon ‘;’, question mark ‘؟’, percent sign ‘٪’, ornate left parenthesis ‘﴿’, ornate right parenthesis ‘﴾’
Thai	bullet ‘●’, ellipsis ‘…’, section ending ‘๑’ (<i>Angkhankhu</i>), chapter ending ‘๒’ (<i>Khomut</i>)
Greek	question mark ‘?’
French	left quotation mark ‘«’, right quotation mark ‘»’
Ethiopic	wordspace ‘፡’, full stop ‘።’, comma ‘፣’, semicolon ‘፥’, question mark ‘፧’, paragraph separator ‘፨’
Spanish	inverted question mark ‘¿’, inverted exclamation mark ‘¡’
Chinese, Japanese	comma ‘、’, period ‘。’, ditto mark ‘〃’
Tibetan	brackets ‘།’ (<i>Gug rtags gyon</i> mark) and ‘༎’ (<i>Gug rtags gyas</i> mark), list enumerator ‘༐’, topic separator ‘༑’

Figure 14. Some examples of punctuation and Marks (Savourel, 2001).

- **Word separation:** spaces are not always used to separate words. For example, Korean, Japanese and Chinese languages do not use spaces to separate words, while in other languages special characters are used. Figure 15 shows an example of Thai language, in which words are not separated by spaces.

กลับไปอ่านอัลกรอาน/สงสรเกียรติทูตงานศพ

Figure 15. A sentence in Thai language where words are not separated by spaces.

- **Digits:** deeply different are the representation of numbers in the various languages. Some of them are presented in Figure 16.

Script	Digits									
Western	0	1	2	3	4	5	6	7	8	9
Arabic-Indic	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Eastern Arabic	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Bengali	০	১	২	৩	৪	৫	৬	৭	৮	৯
Devanagari	०	१	२	३	४	५	६	७	८	९
Lao	໐	໑	໒	໓	໔	໕	໖	໗	໘	໙
Gurmukhi	੦	੧	੨	੩	੪	੫	੬	੭	੮	੯
Gujarati	૦	૧	૨	૩	૪	૫	૬	૭	૮	૯
Oriya	୦	୧	୨	୩	୪	୫	୬	୭	୮	୯
Telugu	౦	౧	౨	౩	౪	౫	౬	౭	౮	౯
Kannada	೦	೧	೨	೩	೪	೫	೬	೭	೮	೯
Malayalam	൦	൧	൨	൩	൪	൫	൬	൭	൮	൯
Tibetan	༠	༡	༢	༣	༤	༥	༦	༧	༨	༩
Thai	๐	๑	๒	๓	๔	๕	๖	๗	๘	๙

Figure 16. The representations of numbers in various languages (Savourel, 2001).

- **Writing direction:** languages like Hebrew and Arabic have right-to-left writing direction. Other languages, like Chinese, Korean and Japanese are written from top to bottom, with columns ordered from right to left. The different writing directions affect the way in which the elements in an interface should be presented and ordered. Figure 17 represents two versions of the Wikipedia home page, one localized to English language and one to Arabic language.
- **Formatting styles:** typography choices are also affected by the languages. For example, while Westerns use bold text to emphasize, Chinese or Japanese use dots or accent-like symbols above or below characters. Layout also differs, in that for

example in Arabic, justification of text is done with elongated connections between letters.



(a)



(b)

Figure 17. Two versions of Wikipedia: (a) English and (b) Arabic. The organization of the page follows the writing direction of the language (from left to right for English and from right to left for Arabic).

- **Character shapes:** in some languages, letters could have different forms in lowercase or uppercase. As an example, letter sigma in Greek is represented as 'σ' or 'ς' (when in terminal position), while in uppercase both have the form 'Σ'.
- **Sort order and case:** ideographic-based languages order their characters by phonetic, by stroke count, by radical, etc. Languages that use Roman alphabets

order letters in different ways. For example, in Danish and Norwegian languages, ‘æ’ comes after ‘z’ and ‘ø’ comes after ‘æ’. Another aspect related to sort order, is uppercasing and lowercasing. For example, in German ‘ß’ is uppercased as ‘SS’.

- **Date and time formatting:** different representation format of time could be used, like 12-hour or 24-hour formats. Also the order in which the elements of a date are combined and the characters used as separators can change.

Other aspects that are affected by cultural conventions and that should be considered during the localization process are aesthetic, the use of symbols, the contents, capacity measures, and currencies (Nielsen, 1993; Nielsen, 1996). In fact, as described in this chapter, in order to internationalize software products it is not enough to translate text. Language used for the contents and the interfaces should reflect the values, ethics and morals of the target users (Yeo, 1996).

3.4.4. GILT in practice

The GILT methodological model can be implemented in practice following one of the three methods:

- 1) The software is developed creating different copies, all localized and translated for every locale to be considered (Figure 18). This is the easiest way of proceeding but implies a high workload each time a new locale has to be added and/or every time something has to be changed in the software. Moreover, there is high redundancy of code and this approach does not follow strictly the definition of GILT model given in the previous section.
- 2) The software is developed in two separated parts (Figure 19): a core that is “neutral” in respect to the locales to be used, and a unique file of localization in which all the elements that need to be localized and translated are stored (e.g. text and images). This method follows the definition of GILT model in that the software is internationalized and separated from its elements that are influenced by locale-dependent characteristics. However, in case of a large number of locales, the

unique file of localization is hard to be used since its size would highly increase and because of the consequent latency in accessing and reading it.

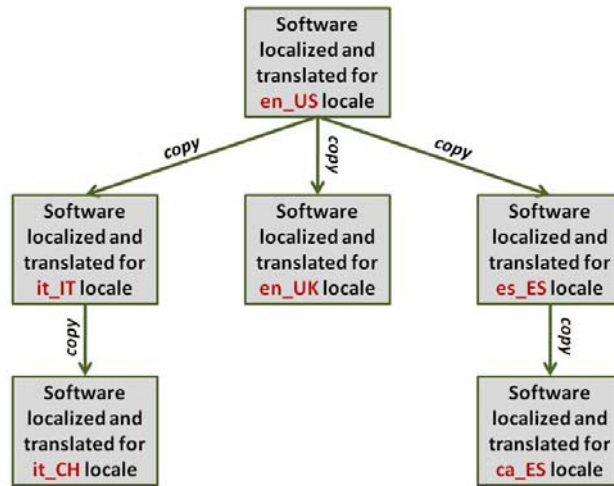


Figure 18. The first method of implementation of the GILT methodological model.

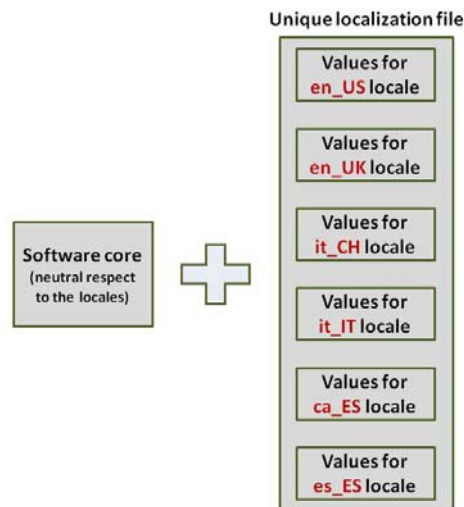


Figure 19. The second method of implementation of the GILT methodological model.

- 3) As in the previous method, the software is separated into two parts: the core and the localization file (Figure 20). However, in this case there are as many localization files as the locales considered. This means that adding a new locale is simpler because it implies the creation of just one more file while the existing ones do not have to be modified.

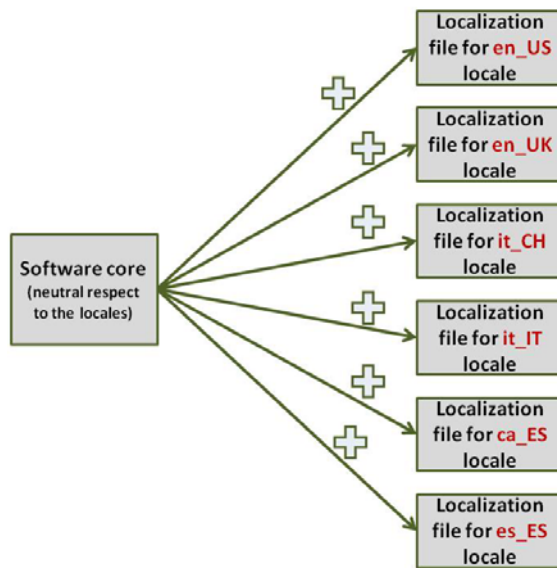


Figure 20. The third method of implementation of the GILT methodological model.

This thesis presents a concrete implementation of what theoretically defined by Yeo (1996) with the CUI concept. The architecture described in Part IV, allows in fact the development of internationalized systems that are localizable according to user's culture, role that the user plays in a context, and to the digital platform in use.

Chapter 4:

Challenges and thesis contributions

This chapter presents the challenges that arise from the context described by the previous chapters in this first part of the thesis. In particular, the contributions of the thesis are highlighted.

4.1. Challenges

From the context described in the previous chapters, three main challenges emerge and are addressed by the architecture proposed in this thesis.

4.1.1. System customization

What presented in Chapter 1 about EUD and in Chapter 3 about global communities, internationalization and localization, leads to the necessity of designing and developing software systems adaptable to different aspects of the user's interaction. In particular, the literature review on localization, highlights the need of localize software systems to the users' culture, to the role played by the users in a context and to the digital platform they use to access the system.

In order to face these challenges, the design process should be characterized by the use of programming techniques that support the customization. This means that the whole

program structure should be easily modifiable and extendable and that at any time new localization processes and/or new features should be added easily.

4.1.2. System evolution

What presented in Chapter 1 about end-user development and in Chapter 2 about cultures of participation opens the discussion about how to support the co-evolution of users and systems and the involvement of the end users in the process.

In order to face this challenge, the design and development of software systems should be considered as a never-ending process that leads to the creation of perpetual beta systems and no more towards an approach based on successive releases.

4.1.3. Communication

As emerges from what presented in the first three chapters of this thesis, communication among all the stakeholders involved in a collaborative project is fundamental. As to end-user development, communication is important because a correct involvement of many domain experts in a single project should be characterized by a continuous bargain between all of them. As to cultures of participation, the problems related to the communication gap among the various communities of practice should be fixed enhancing the communication tools available. As to global communities, the issues related to multicultural work context have to be faced by allowing the stakeholders to collaborate expressing their opinions with the support of localized tools.

4.2. Thesis contributions

The research activity developed is framed in the field of computer semiotics and in particular of semiotic engineering, a discipline that views interactive software applications as metacommunication artifacts.

The line of research follows an experimental approach, in that it starts with the observation and the study of real cases and it continues with the specification of the methodology and the definition of general models.

The general methodology of design research followed is the one described in (Vaishnavi and Kuechler, 2004; Nunamaker et al., 1991) illustrated in Figure 21.

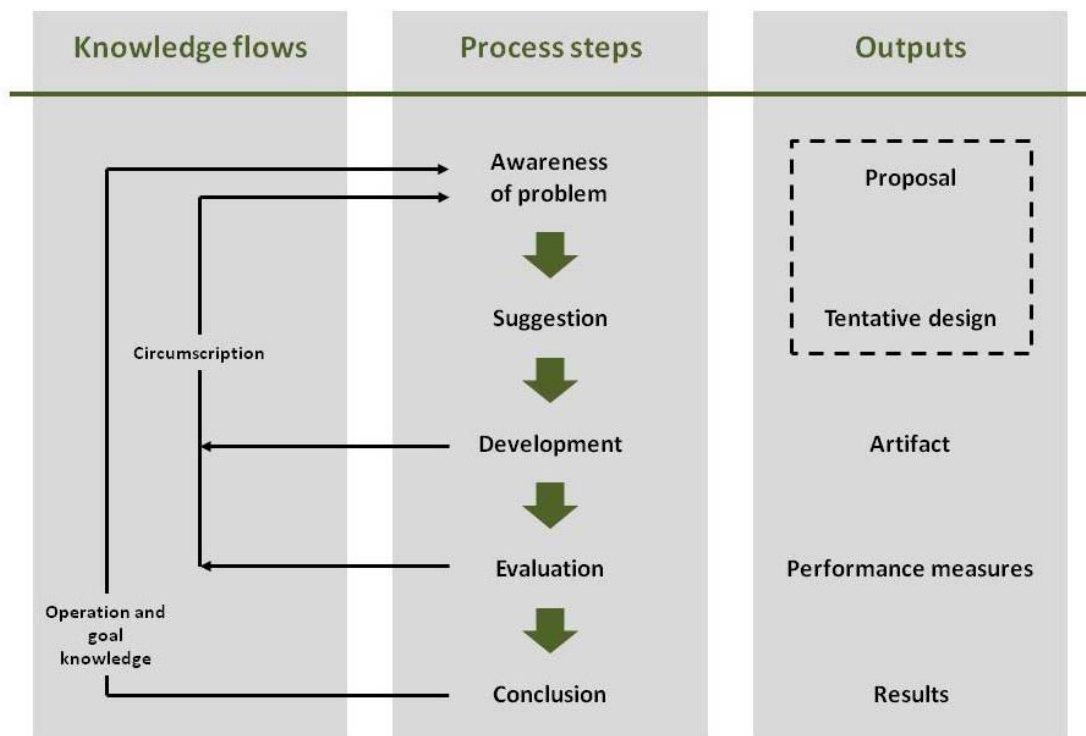


Figure 21. A general methodology of design research. Adapted from (Vaishnavi and Kuechler, 2004).

A set of problems arise from the study of the context and of real cases in different domains. From the problems and the literature review the three challenges described in the previous section emerged and lead to the preparation of a proposal, to tentative design and to suggestion step. Up to now, I have analyzed several cases in different domains (e.g. medical, touristic, constructions) and for them a set of interactive systems has been designed and developed taking into account the needs of the users, their culture, mother tongue and writing systems, as well as the role they play in the community that they belong to and the digital platform and the technologies they use to access the systems. These

systems are illustrated in Chapter 9. Different evaluations have been performed on the systems and qualitative analyses of data have been computed.

The research activity developed during the PhD years started from a generalization of the experience developed before at the Italian National Research Council.

The focus of this research is on the definition of an architecture for end-user development supporting global communities that implements a semiotic model on end-user development. The architecture supports the design and development of interactive systems that allow end users, who are not expert in computer science, to participate in the creative design and development of their own systems based on their user experience. These end users are considered knowledge workers using interactive systems which support them in the access, creation and management of shared knowledge bases and in the development of tools to support these activities specialized for their needs. The approach I follow is collaborative, in that the creative design and development of interactive systems requires the participation of different stakeholders, e.g. software engineers, HCI experts and domain experts. This collaboration requires the stakeholders to make their expertise and reasoning explicit, by externalizing them as multimodal documents to be shared and discussed in the community.

The next part of the thesis will illustrate all the case studies developed in the last years. The various contexts are presented and the requests that arose from them are highlighted. In chapter 9 the case studies will be described from the implementation point of view, explaining how the architecture has been applied to them.

Part II: Case studies

Chapter 5: Case studies

This chapter presents the case studies developed in the last 5 years. They regard different application fields, e.g. factory automation, building and construction, tourism and cultural heritage. My research follows a pragmatic approach, in that it starts from the observation of concrete cases, to arrive to a practical implementation of virtual interactive systems. Each case study influenced the definition of the semiotic model and of the architecture presented in this thesis, by pointing out challenges and necessities to be faced. The implementation of the cases presented in this chapter is described in Part IV, and the results of the evaluation of one of them is presented in Part V.

5.1. Factory automation

This first case study resulted from the collaboration of Università degli Studi di Milano, Università degli Studi di Bari, Università degli Studi di Brescia and ETA Consulting, a company that produces systems for factory automation.

The company needed to create systems for factory automation usable for their client companies, easy to be learned and to be used and that are customizable for experts in factory automation but not in computer science (Costabile et al., 2007b; Costabile et al., 2008a; Fogli and Piccinno, 2005; Barricelli et al., 2009c; Zhu et al., 2010b). Furthermore, ETA Consulting needed to develop software tools to support its personnel in development,

testing and maintenance of their factory automation systems. The personnel in the company was organized in different communities of practice, each one having different skills and responsibility and needing to perform different tasks supported by software tools.

The goal of the intervention was to design and develop interactive environments each one specific for a community of practice. The expected scenario is depicted in Figure 22.

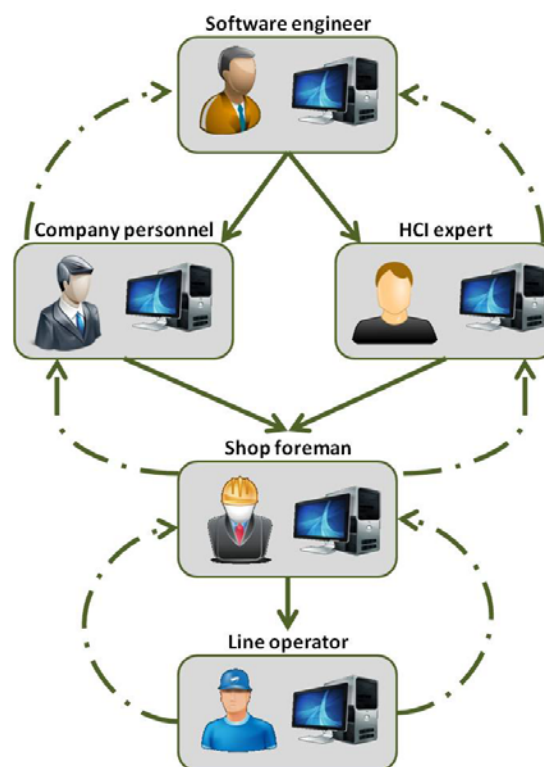


Figure 22. The expected scenario of the factory automation case study. The downward arrows represent the exchange of the software environments among the stakeholders, while the upward arrows represent the annotations sent among them.

Several CoPs are involved in the company: software engineers, HCI experts, company personnel, shop foremen, line operators. The software engineers are in charge of designing, developing and maintaining all the environments created for the other members of the company. HCI experts, together with some representatives of the company personnel (domain experts) are in charge of creating the entities to be managed by the shop foreman

and the line operators at the client company. The shop foreman, through her/his environment creates environments for the line operators.

In this case study, the HCI experts and the various end users (company personnel) are supported by the use of software environments in acting as end-user developers. The participation of all the members of the different CoPs is granted by the use of annotation. Through these tools, two types of documents are exchanged among all the stakeholders: 1) the environments that are developed by the software engineers, and the ones developed by the shop foreman for the line operators, and 2) the annotations about usability problems, new user needs or proposed improvements.

5.2. Building construction

This case study is the first to which I actively collaborated. It regards the building construction sector and was developed in the frame of the COL (Cantiere On Line) Project (Fogli et al., 2005), funded by the Italian Government and carried out by the National Research Council (Institute for Construction Technologies), the Department of Information and Communication of Università degli Studi di Milano, two SMEs in ICT field, and two enterprises in the building construction sector.

In Italy, the building construction field is characterized by the presence of many small and medium companies who deal with geographically distributed yards and manage large amount of data and information.

In this context, different users having specific roles, experiences, and skills, belonging to different communities of practice, collaborate on the same project forming therefore a community of interest. In the case study, two principal work places are considered: a) the firm technical office and b) the building yard.

Many problems arise in this situation, because the management and update procedure of technical drawings regards different versions of the same documents: electronic in the technical office and paper-based on the yard. In particular, paper-based technical drawings are most of the time updated in the yard but more of the half of these updating are not

reported to the office (Fogli et al., 2005). This results in high cost in the construction process and in building maintenance.

The goal of intervention was the overcoming of these problems. The expected scenario is described in Figure 23.

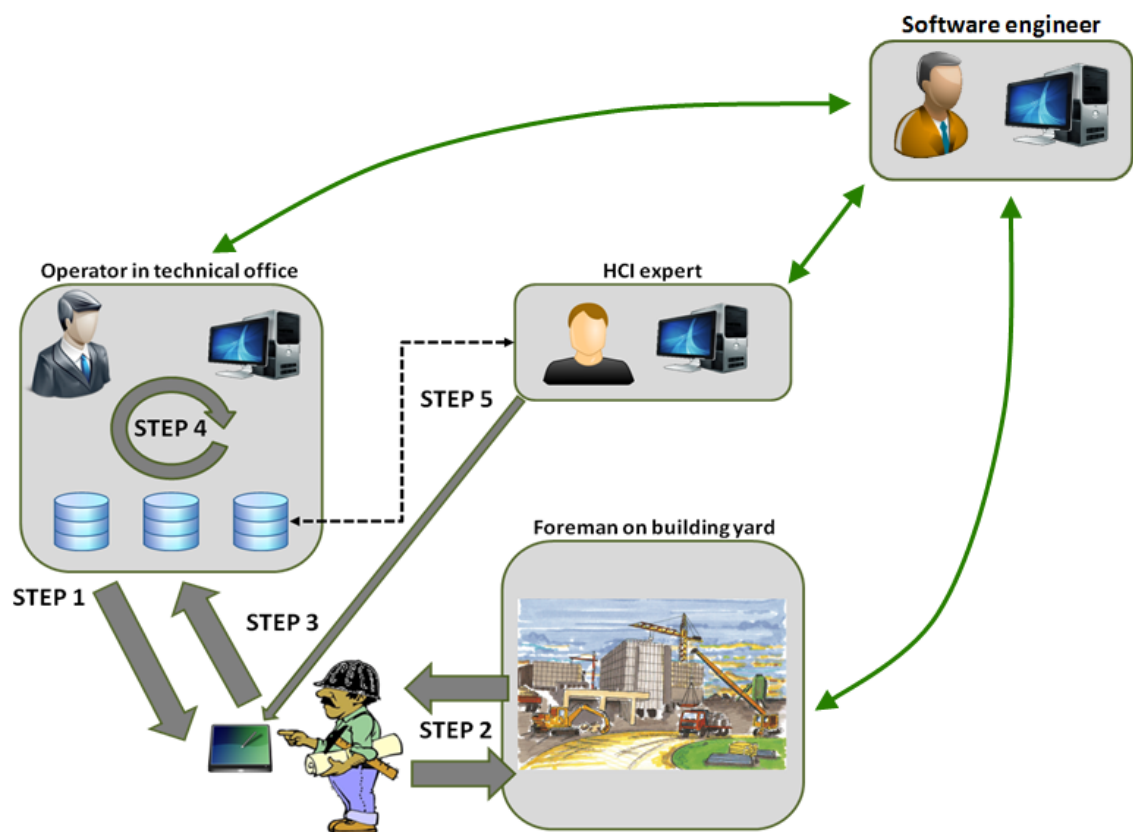


Figure 23. The building construction sector case study context.

In the firm technical office all documents, both electronic and paper-based, are stored in a shared repository. This repository represents the knowledge base of the firm and is constantly updated and accessed by the member of the firm. On the building yard, the foreman uses an interactive system running on a mobile device with a small display in which the same document visualized by the yard operator is represented. Both the operator and the foreman report their activities by annotating the documents on the screen. Before starting to work in the building yard, the foreman loads on the mobile devices a copy of the electronic documents s/he needs, i.e. the technical drawings of a floor of a building (step

1). These drawings will be small-sized by means of minimal graphics techniques, which permit to determine only the signs needed to understand and annotate the drawings. The foreman annotates the changes in the drawings and saves the changes on the mobile device. Moreover, s/he can annotate also her/his usage difficulties, highlighting the tools which generate them (step 2). Once the foreman is done with the work on the building yard, s/he can send the annotations to the technical office, via a wireless connection or uploading them using a wired connection by a desktop pc (step 3). In both cases, the annotations are sent to and stored into a temporary archive. An operator on a desktop computer in the technical office updates the version of the electronic documents that are stored in the shared knowledge base and saves them in the archive as annotated documents (step 4). Usage difficulty annotations are stored in a different archive, to be used by HCI experts. HCI experts, by using dedicated virtual environments, apply the changes requested by the users in order to meet their needs (step 5). In this scenario, the software engineers design, develop, and maintain the environments created for the HCI expert and for the operator in the technical office. Also in this case, annotation tools are used to allow all the stakeholders to communicate their difficulties and requests to the software engineers.

In this context, the different stakeholders who take part in the process are presented, as well as their activity as members of the community of interest. The scenario described above highlight how several interfaces and interaction techniques have to be designed and evaluated: those that will support the foremen in the yard, those which will support the operators in the technical office to update the archives and those which allow the building surveyors, the building designers, the building administrators to access and manage the data in the shared repositories.

In this case study, the HCI expert together with the operator of the technical office (the domain expert), represent two end-user developers who design and develop the software environment for the foreman working on the building yard.

The participation of all the members of the organization, that is a community of interest, is supported by the use of the annotation tool and by the accessibility of shared knowledge bases.

5.3. Medical collaboration

Several studies were performed in the medical field, in order to support the collaboration among different physicians and other members of the health systems that work together to achieve a diagnosis (Costabile et al., 2006b; Costabile et al., 2007a; Costabile et al., 2007b).

This specific case study started from the results of a project developed by Università degli Studi di Bari, the neurology department of “Giovanni XXIII” Paediatric Hospital of Bari, Università degli Studi di Brescia and National research council. In this project, different physicians with various specialties were involved, e.g. neurologists and neuroradiologists. The physicians were observed during their analysis of clinical cases and in the generation of the diagnosis. The field study performed, studied the ambient and organization factors influencing the work of the physicians, the flow of activities, the end users involved and their main tasks. What emerged is that in serious and difficult cases, physicians exchange consultations with other physicians to better study the pathology of the patient. For example, the neurologist might refer to a neuroradiologist for a more detailed analysis of the patient Magnetic Resonance Images (MRIs) and the neuroradiologist may need to consult another neuroradiologists specialized in that particular pathology.

Very often, consultations occur during face-to-face meetings in the neurology department where consultants may come from other hospitals. The cases are discussed one at a time and always with the same procedure: a neurologist chooses a case, describe the most relevant data about the clinical history of the patient and gives the MRIs to the neuroradiologist. The neuroradiologist chooses 3 or 4 images of interest and put them on the diaphanoscope to analyze them. Neurologists and neuroradiologists exchange information in order to clarify possible doubts and converge to an agreed opinion. At the end, the diagnosis on which the specialists agreed is written on the patient record and the next clinical case is considered.

The goal of intervention was the realization of the scenario depicted in Figure 24.

In this scenario, the physicians are provided with software environments and tools which are both usable and tailorable to their needs. For example, if a neurologist needs to consult a neuroradiologist, s/he requests it by annotating the MRI s/he is analyzing. The annotation is composed by two parts, first the question to be asked to the colleague, and then the description which summarizes information associated to the question. The annotation tool could be used also to request changes to the software environment. These annotations are sent to the HCI experts who are in charge of adapting the environments to the physicians' needs and requests.

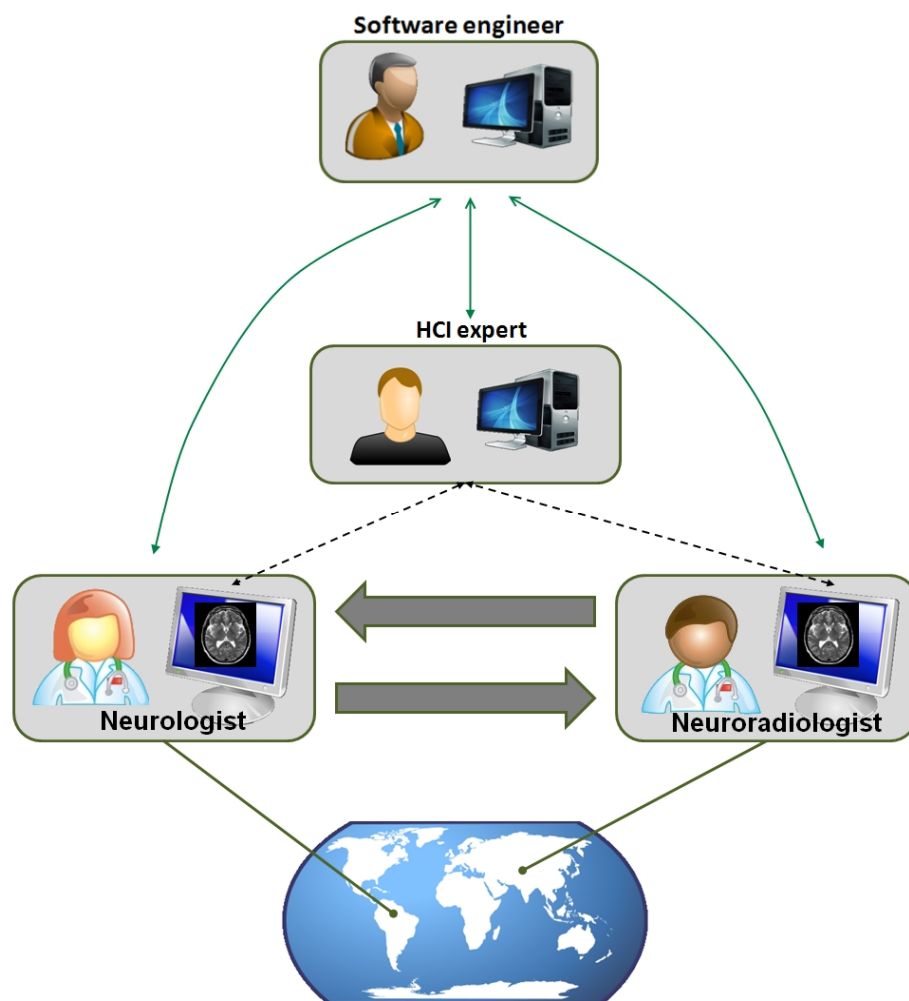


Figure 24. The medical collaborative scenario in which two physicians collaborate remotely to the same MRI analysis in order to reach a diagnosis. The HCI expert applies the changes on the environments requested by the physicians.

In this case study, the HCI expert is an end-user developer who designs and develops the software environment for the collaborating physicians.

The environments used in this context by the HCI expert and by some representatives of the physicians are designed, developed and maintained by software engineers who can receive requests and suggestions by all the stakeholders by using the annotation tool.

The physicians and the HCI expert form a community of interest, and their collaboration is supported by the use of the annotation tool and by the accessibility of a shared knowledge base.

In order to support a global medical collaborative context, the need of designing internationalized software applications for remote and asynchronous work arises. In this scenario, particular attention is given to processing, indexing, and retrieval of large collections of e-documents because, according to different cultures and systems of signs, the shared knowledge base should be made accessible by and usable for them.

5.4. Tourism and cultural heritage

This case study is developed in the frame of the SCV (Sistema Culturale Valchiavenna) project. The project is funded by Fondazione Cariplo and is carried out by the Territorial Community of Valchiavenna valley together with Università degli Studi di Milano and other partners (Marcante and Parasiliti Provenza, 2008; Barricelli et al., 2009a; Zhu et al., 2010a; Zhu et al., 2010b).

The SCV project aims at coordinating, planning and promoting tangible and intangible assets relating to artistic, historical and cultural heritage of a well-defined geographic area in the north of Italy: the Valchiavenna valley.

The final goal of the intervention in this case study is the exploitation of interdisciplinary capabilities and experiences developed by lecturers and researchers of several institutions to show how the use of digital communication methods can enable experts from different disciplines to work together for creating a continuously updated knowledge base. The

contents stored in the knowledge base are immediately available for dissemination and for guiding tourists that wish to discover the area. The need of designing and developing new contents and new techniques to manage contents has brought towards the creation of a new knowledge model in the field of the safeguard, valorization and dissemination of cultural assets. Several communities of practice are involved in the SCV project, e.g. historians, economists, and geographers.

The project aims at providing them with tools for knowledge accumulation and dissemination. At the same time, also another community is considered, the one of the tourists, that aims at accessing the domain expert knowledge base and at collaboratively commenting the maps and the points of interest of the Valchiavenna valley.

The expected scenario is depicted in Figure 25.

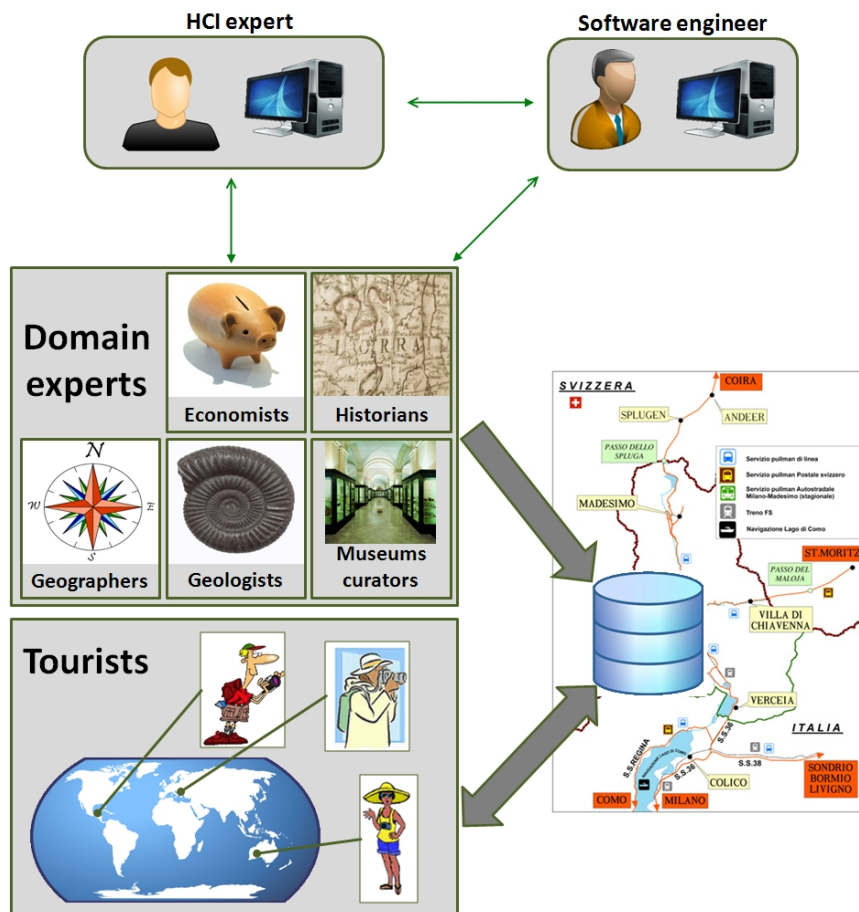


Figure 25. The scenario in the SCV project.

The domain expert, supported by other members of the community of interest (e.g. HCI experts), design and develop not only the knowledge base to be shared with the tourists, but also the tourists' software environment, by choosing the tools to be provided to them.

In this case, the domain experts and the HCI experts are end-user developers, and as in the previous cases, the participation of all the stakeholders in the community is supported by the annotation tool and by the shared knowledge base.

The environments used by the HCI expert and by the domain experts are designed, developed and maintained by software engineers who receive requests and suggestions by all the stakeholders through the use of annotation tools.

As in the medical case, there is the need to realize a system that is internationalized and localizable in order to satisfy a global audience.

5.5. Workflow management

The case study in workflow management context, is developed in the frame of a collaboration between the Computer Semiotics Laboratory at the Department of Information and Communication of Università degli Studi di Milano and the Institute of Construction Technologies of the National Research Council (Barricelli et al., 2009d; Scala, 2010; Barricelli et al., 2010b; Barricelli et al., 2010c; Barricelli et al., 2010d).

It stems from the experiences developed in the last years, with two research units of the Institute of Construction Technologies of the National Research Council that act as certification institutions. The two units need to semi-automatize the certification processes that were originally carried out manually. In particular, one of the two units case is considered in this thesis, that is the one that works on accord transport perissable (ATP) certification of the vehicles.

The ATP certification is a process consisting of different tests made on vehicles used in the cold chain industry. In particular, the process aims at calculating the global thermal swap coefficient in insulated vehicles and cisterns and at the production of minutes that certifies

the vehicles' conformity to the existing legislation. Conceptually, the process is structured in various concurrent activities where different kinds of data are collected and elaborated to produce the certification documentation by the following software components:

- a) harvesting of general information about the vehicle, manually inserted;
- b) harvesting of specific information coming from measurements previously made on the vehicle and automatically acquired;
- c) elaboration of the information acquired and production of new information;
- d) production of the certification documentation.

What is needed by this context is a system that supports the members of the certification institution in design, execute, and check the execution of a workflow. A workflow is a model to formalize a complex work process for further assessment and manipulation (e.g. for optimization or iteration of specific sequences of tasks) (Miller, 2000). A work process is described as a pattern of simple tasks, information flow, resources, and roles. Workflow management of cooperative groups of people is supported by software tools to share information, manage communication, schedule and assign tasks to the co-workers. Two communities of practice participate actively to the life cycle of a workflow. The first community is the one of the Workflow Designers, which are domain experts, belonging to the institution, not expert in computer science, in charge of designing the current workflow and of supervising its correct execution. The second community is the one of the Workflow Operators, which belong to the same unit of the Workflow Designer, not expert in computer science, in charge of executing the workflow process.

The scenario that is expected is depicted in Figure 26.

In this case study, the workflow designer (supported by HCI experts) is an end-user developer. The environment for the Workflow designer and for the HCI experts are designed, developed and maintained by the software engineer who is also able to receive suggestions and modification requests by them through the annotation tool. Moreover, annotation is used to support the participation of all the stakeholders in the community and the creation and management of a shared knowledge base.

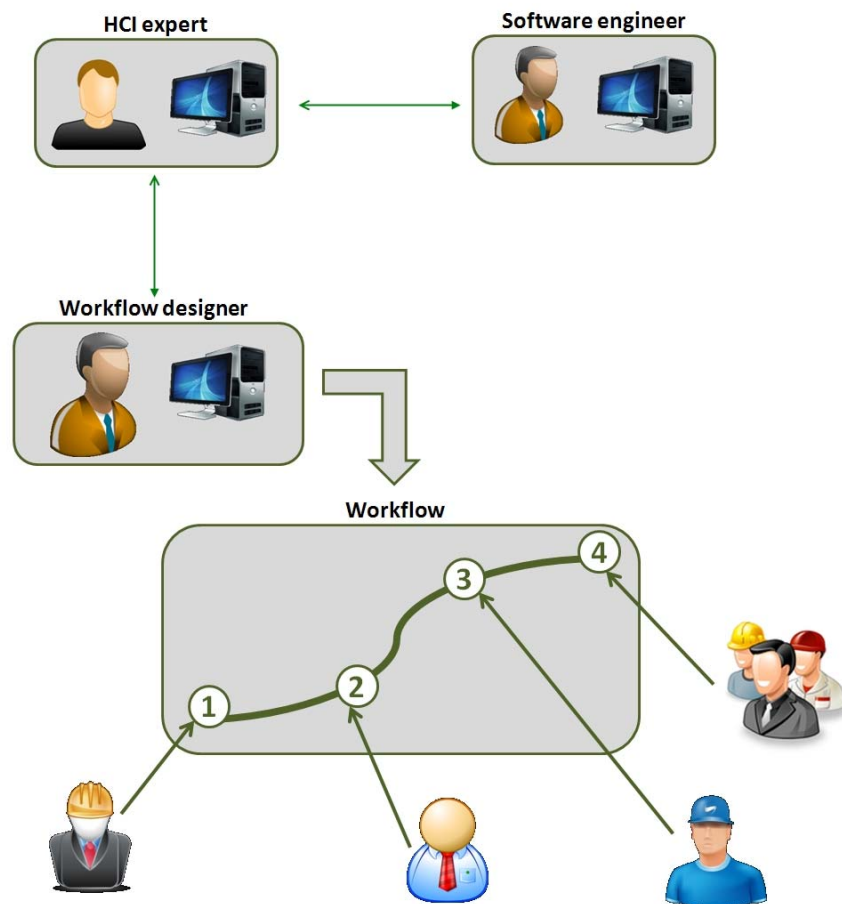


Figure 26. The workflow designer designs the workflow to be followed by the operators.

In this chapter all the case studies observed and studied during the last year have been presented. In Part IV, the architecture presented in this thesis and its implementation for the different case studies is described. Part V presents the results of the evaluation of the tourism and cultural heritage case study.

**Part III:
A semiotic model
for end-user
development**

Chapter 6:

Semiotic view on digital communication

This chapter presents a semiotic view on the communication process, first by analyzing oral and written modalities, then by describing the digital communication process and the phenomena by which it is affected. Computer semiotics approach to HCI and semiotic levels of analysis of the digital communication process are also discussed.

6.1. Semiotic view on communication process

The semiotic view on communication processes emerges from the research performed taking as starting point the work of Ladislav Tondl (1981). His definition has been evolved in (Marcante and Mussio, 2006) by making explicit the role of human actors in the communication process.

A communication process presupposes the existence of:

1. an environment and the phenomena that take place in it and that generate events that change the state of the environment (e.g. voices, written messages, feedback in interaction with a software system);
2. at least two communicant systems in the environment which:
 - are able to perceive events generated by the phenomena that take place in the environment (i.e. changes in the state of the environment);

- are able to assign a relation among the events they perceive and their view of the phenomena in the environments;
- are able to interpret the events they perceive and to establish an assignment system (i.e. the set of relations among the perceived events and their view of the phenomena in the environment);
- are able to generate events that affect the state of the environment.

3. a system of communication means constituted by:

- a finite set of events $E = \{\alpha_1, \dots, \alpha_n\}$ which can be transmitted along a channel;
- a channel to transmit the events;
- tools to support the communicants in coding (decoding) their views of the environments into events to be transmitted along the channel.

Coding means to translate a concept into events by means of human organs (e.g. voice, hands), physical systems (e.g. telephone handset, pencil, keyboard) or the combination of the two.

Perceiving and decoding means to translate events into concepts using human organs (e.g. eyes, ears, hands, cognitive system) with eventually the support of physical systems (e.g. telephone handset, monitor). The events may be combined together to form expressions: each sequence of events or expressions that is transmittable on the channel, is called a message.

The communication process model is depicted in Figure 27. F_1 and F_2 are two worlds of phenomena perceived by communicant 1 (I_1) and communicant 2 (I_2). α is the message that is exchanged between the two communicants. C stands for coding, while D stands for decoding. These two activities are performed by the two communicants on the messages exchanged. On the basis of the type of communication that takes place, coding and decoding activities may be performed by means of human resources or by means of physical tools or both.

The communicant recognizes the events and associates to them a meaning.

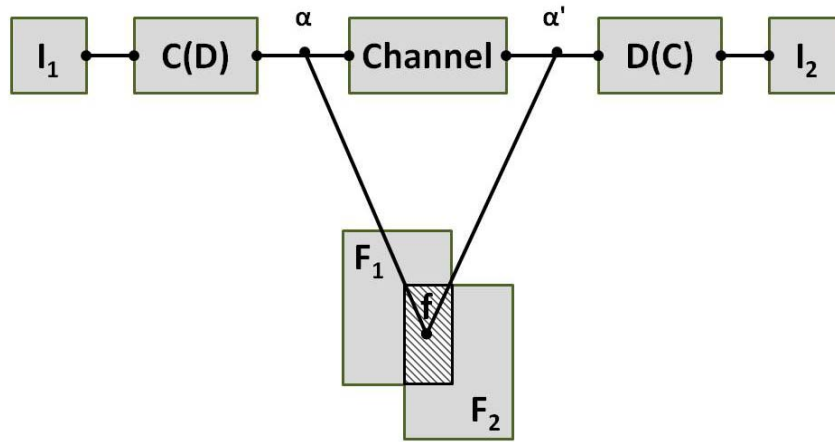


Figure 27. The communication process described in (Tondl, 1981) and evolved in (Marcante and Mussio, 2006).

By assigning a meaning, the communicant performs a semantization process. This means that each communicant may assign different meanings to the same perceived event. When all the communicants involved in a communication process assign a compatible meaning to an event, the semantization process is said successful, because they reach a mutual understanding.

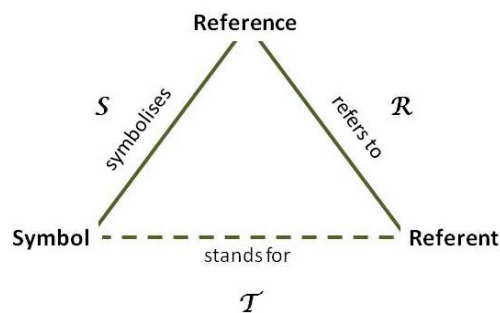


Figure 28. The Ogden and Richards semiotic triangle.

A model that explains how a communicant associates symbols, concepts and elements, perceived in the world of phenomena, is the Ogden and Richards semiotic triangle (Ogden and Richards, 1960).

The semiotic triangle of Ogden and Richards (Figure 28) allows to describe oral and written communication and decontextualizes the symbol from its context of use (Harris, 2000). In fact, symbols are products of the communication process itself and they should be considered as part of social and physical context.

The Ogden and Richards semiotic triangle defines three factors and their relations. The three factors are the reference (i.e. the mental process, the thought), the symbol (e.g. the word), and the referent (the real object to which the communicant refers). The symbol symbolizes the reference and the referent refers to the reference. Between the symbol and the referent there is no direct relation, (in Figure 28 the two factors are connected by a dashed line). The relation between symbol and referent is mediated by the mind of the communicant and is therefore variable and individual.

In the semiotic view on communication adopted in this thesis, the term symbol used by Ogden and Richards is substituted by the term signal, after (de Mauro, 1982). The revised triangle is depicted in Figure 29.

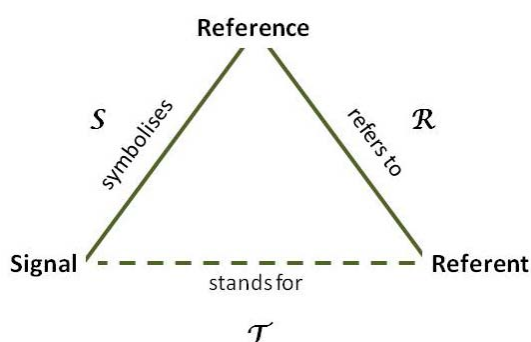


Figure 29. The Ogden and Richards semiotic triangle revised according to de Mauro (1982).

The revised semiotic triangle can be applied to the communication process represented in Figure 27. The two red triangles (1 and 2) represent the semantization processes of the two communicants (1 and 2).

In the case of oral communication, the model in Figure 30 can be applied and the process has the following characteristics:

- The two communicants (I1 and I2) are humans.
- In this case, the tools for coding (C) and decoding (D) are human organs (vocal apparatus and auditory apparatus) and the communicants do not need to use other physical tools.
- The message (α) is a sound.
- The channel conveys the sound.
- The persistence of the message is limited to the duration of the message emission.
- The communication process is synchronous.
- The communication process is situated.
- The communication is bidirectional, in that sender and receiver may exchange their role.

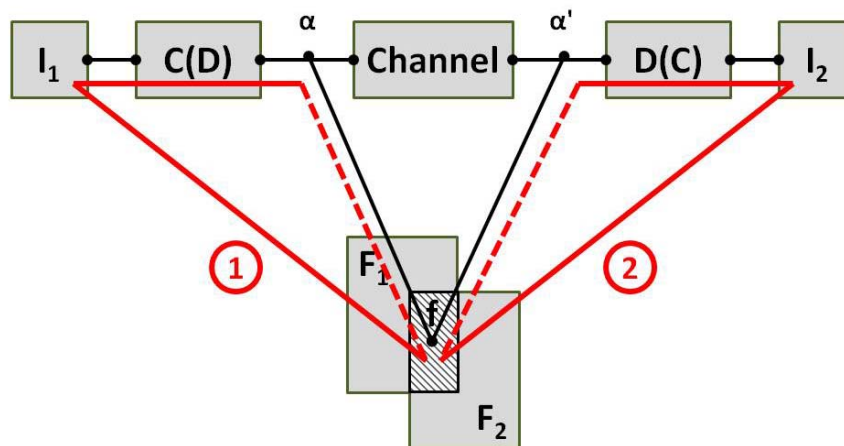


Figure 30. The two communicants reach a common understanding if they assign compatible meanings to the same signal. The two red triangles (denoted by 1 and 2) represent the semantization processes of the two communicants.

In the case of written communication, the model in Figure 30 can be applied as well and the process presents the following elements:

- The two communicants (I1 and I2) are humans.
- In this case, the tools for coding (C) and decoding (D) are both human organs and external tool, (e.g. pens, keyboards, pencils).
- The message (α) is a medium modified by the use of a tool.
- The channel includes the visual apparatus.

- The persistence of the message is unlimited: the medium modified records and makes perceivable the message in a perpetual way unless medium damage.
- The communication process is asynchronous.
- The communication process is performed without the presence of the author of the message.
- The communication is unidirectional and not interactive.

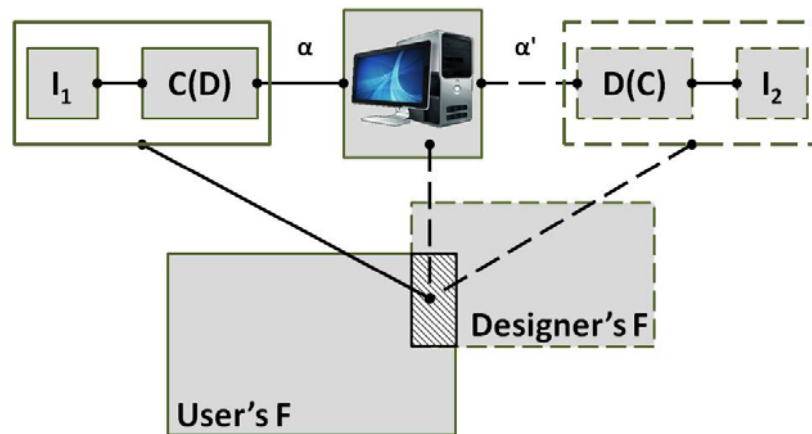


Figure 31. The digital communication process model. The designer of the software system is not present during the process of interaction between the user and the system s/he developed.

The digital communication process is characterized differently (Figure 31):

- The two communicants are a user (I_1) and a software system designer (I_2), but the interaction takes place between the user and the system developed by the designer, that is the message sent on the channel.
- The message (α) is virtual, transient, dynamic, and multimodal.
- The process takes place in presence of user and computer but with absence of the software system designer (author of the message).
- The tools for coding (C) and decoding (D) are both human organs and software programs.
- The channel is multimodal and managed by software programs.

- The persistence of the message is a) perpetual for its internal form and b) transient for its materialization (exists as long as a software program interprets and materializes it)
- The communication process between human and computer is synchronous.

The three types of communication – oral, written, and digital – can be represented with their three semiotic triangles joined together in one unique triangle, like in Figure 32. Even if the communication can be based on three different channels and types of signals, the relations among them are not completely distinct and are, to some extent, dependent to each other.

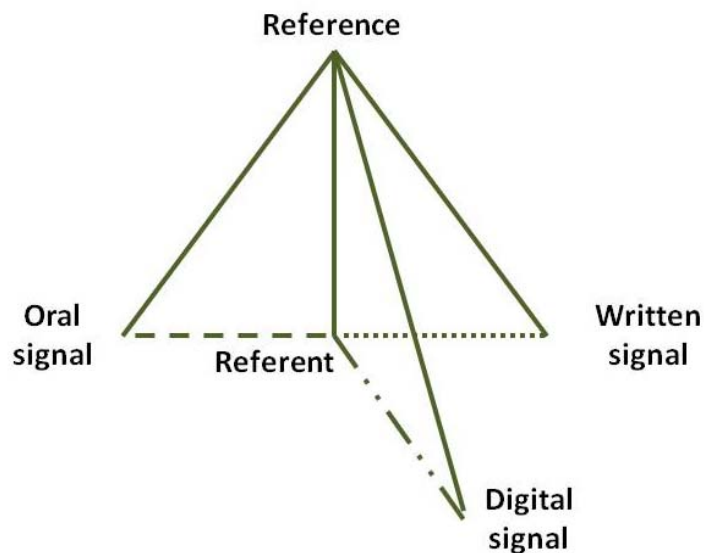


Figure 32. The semiotic triangle that joins oral, written, and digital signal interpretation.
Adapted from (Mussio, 2009; Valtolina, 2010).

6.2. Semiotic view on digital communication process

The digital communication process is defined in (Bottoni et al., 1999) as:

- an interaction process based on bidirectional communication between human and computer;

- a syndetic process in which human behavior influences and is influenced by the computer behavior.

This point of view is illustrated in Figure 33 and considers the communication via a WIMP (windows, icons, menus, pointers) interface (Dix et al., 2004). It evolves the model proposed by Hutchins, Hollan and Norman (1986) that focuses of the human's role in interaction process and that identifies in semantic and articulatory distances the primary sources of usability difficulties. In relation to Figure 31, this definition describes just the interaction between the user and the system and does not consider the metacommunication between the user and the designer.

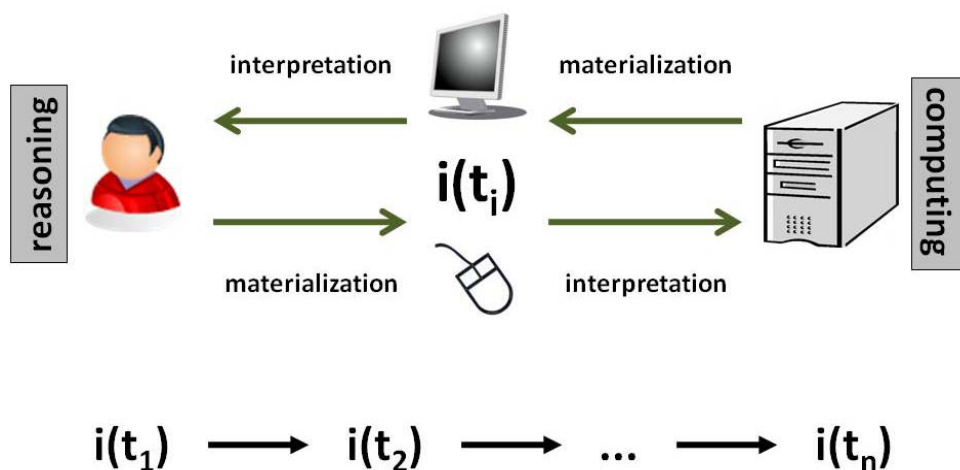


Figure 33. A model of digital communication process.

The interaction between the user and the system is seen therefore as a sequence of cycles:

- the human actor perceives the message appearing on the screen, assigns a meaning to the message, decides what to do next and performs an activity through the use of input devices of the computer.
- The computer captures the inputs sent by the human actors, interprets them, performs the required computations and materializes the results of the computation on the screen. The messages materialized on the screen represent new messages to be perceived and interpreted by the human actors.

The cycle continues until the human actor decides to close it, and this could happen in two cases: 1) the task has been achieved, or 2) the task failed.

6.3. Phenomena affecting the digital communication process

In HCI literature the different phenomena that affect the digital communication process are often reported. The phenomena are observed and studied from many point of views, considering usability, software engineering and software development (Costabile et al., 2006b; Piccinno, 2004).

In (Costabile et al., 2006b) five of these phenomena have been presented and discussed: 1) the communication gap between users and designers, 2) the user diversity, 3) the technological grain, 4) the co-evolution of system and users, and 5) the tacit knowledge and the implicit information.

6.3.1. Communication gap

The communication process is constituted by a sequence of materializations and interpretations of messages that take place at successive instants of time. For each message, two interpretations are performed (Berners-Lee et al., 2001):

- The user's interpretation that depends on her/his culture and role s/he performs in the context (experiences, skills).
- The system's interpretation that depends on the program that assigns a computational meaning to the message.

The system's interpretation of a message reflects the system's designer point of view. System's designers and users have different cultures, knowledge, they reason following different approaches and have different mental models. They reason differently, in that users have a heuristic approach, while designers reason algorithmically. Users' notations reflect their expertise in a specific domain, while software designers use computer science notations; therefore, a communication gap arises between them (Majhew, 1992). In order

to overcome this communication gap, the designer has to a) study the target users of the system s/he is going to design and develop, b) design a conceptual model of the system (i.e. interaction styles and metaphors) adequate to the target users, c) evaluate the conceptual model, and d) adopt an incremental prototyping technique in order to be able to check the validity of the ongoing design and development.

6.3.2. User diversity

According to (Schön, 1983), the end users act as competent practitioners, in that “they exhibit a kind of knowing in practice, most of which is tacit” and they “reveal a capacity for reflection on their intuitive knowing in the midst of action and sometimes use this capacity to cope with the unique, uncertain, and conflicted situations of practice” (Piccinno, 2004). The end users reason on the task at hand and communicate with each other by exchanging documents and using specific notations and dialects related to the context in which they operate. User diversity is strictly related to the achievement of global communities into collaborative design process (described in the first part of this thesis). However, user diversity phenomenon may arise even within the same community, because it can be related to specific physical or cognitive abilities and to the role played by the users in the work context. To face this phenomenon, the system designer has to support the symmetry of ignorance among the members of the different CoPs, by offering tools that are adaptable to users’ needs and habits.

6.3.3. Technological grain

Aroni et al. (2002) defined the “grain” as the tendency of a system to force the users in behaving and reasoning in ways that differ from the ways in which they usually behave and reason. This phenomenon is strictly dependent to choices made by the system designer that makes users following strategies that are to her/him apparently easily executable, but that may be not optimal for them (Piccinno, 2004). The technological grain points out the tendency of a tool to push its users towards certain behaviors (Dix et al., 1998).

The consequences for the system designer, affect in particular the definition of the conceptual model. The designer has in fact to specify precisely each aspect of the interaction style proposed and each designed tool has to be evaluated directly involving the users.

6.3.4. Co-evolution

As already discussed in the first part of the thesis, Nielsen (1993) describes the co-evolution phenomenon stating that

*using the system changes the users, and as they change they will use the system
in new ways*

Users evolve in using a system, and this evolution forces the system designers to adapt the system to the evolved users and to their organizations and environments (Bourguin et al., 2001; Carroll and Rosson, 1992; Arondi et al., 2002; Costabile et al., 2006a).

Figure 34 shows the interaction and co-evolution model (ICE model) (Costabile et al., 2006a). The model describes HCI as a cyclic process, in which the user and the interactive system communicate by materializing and interpreting a sequence of messages (the images on the screen in visual interaction) at successive points in time - $i(t_0)$, $i(t_1)$, ..., $i(t_n)$.

Even if software artifacts are developed aiming at supporting some user tasks, they may also suggest new possible tasks to be performed with them. Therefore, in order to fulfill the end user needs, the artifacts have to be evolved. This task-artifact cycle is denoted as “cycle 1” in Figure 34 and shows a first co-evolution cycle. On the other hand, the advances of technology offer to the computer scientists new possibilities for interactive systems improvement once they are already in use. This situation leads to new interaction possibilities that might change end users working habits. In Figure 34, another evolution

cycle is identified (“cycle 2”): the end user social and organizational context is evolving in time and this requires new tasks and/or different possibilities for task performance. Therefore, technology and social and organizational contexts repeatedly affect each other.

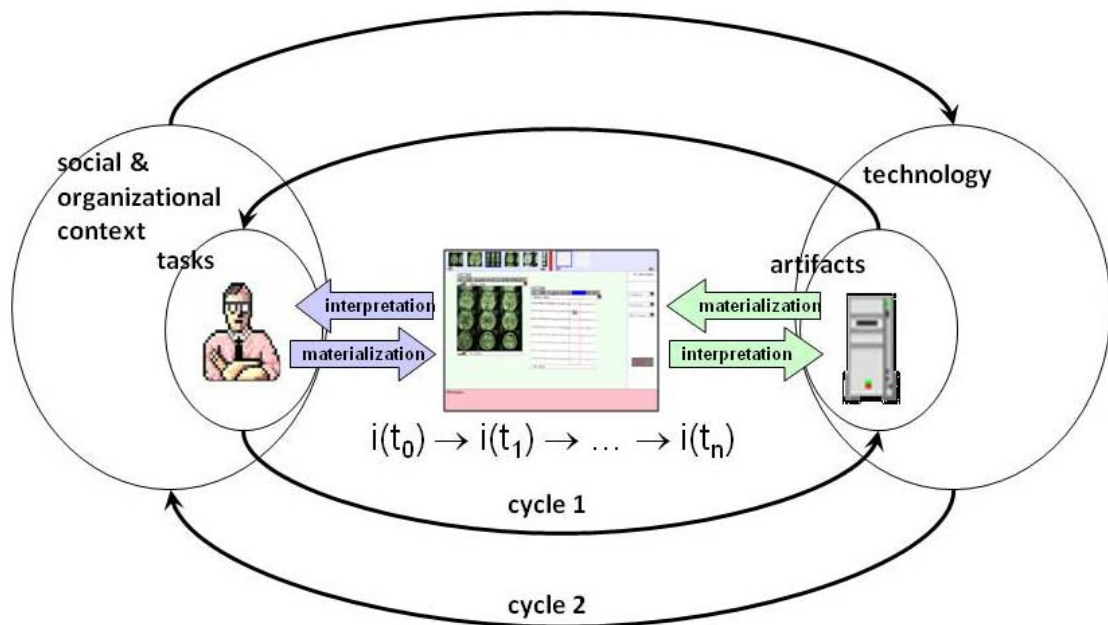


Figure 34. The co-evolution phenomenon (Costabile et al., 2006a).

Lehman et al. (1997) provides a classification of software programs in order to explain how software evolution works:

- **S-type programs:** the specification-based (S stands for specification) programs a) have all their properties (functional and non-functional) completely defined in a specification, and b) have as only criterion of their acceptability to the stakeholders the satisfaction of the specification. S-type programs cannot be improved because by definition they are already completely satisfied by their acceptance criteria.
- **P-type programs:** this type of programs is based on the fact that, in order to design a useful and problem-solving program, some compromises between stakeholders' goals are needed. Therefore, P-type programs (P stands for problem) cannot be specified in a formal way, but through an iterative process.

- **E-type programs:** these programs are also called “real world” programs (E stands for evolving). In fact, E-type programs depend by the real world, model social activities, make assumptions about a context, and provide and require services.

The laws of software evolution that Lehman et al. (1997) derived from the observation of the E- type of program, are the following:

- **Continuing change:** E-type programs have to be continually adapted otherwise they become progressively less satisfactory.
- **Increasing complexity:** the complexity of an E-type program increases while the program evolves but the work done to maintain it becomes less and less.
- **Self regulation:** the evolution of E-type programs is self-regulating.
- **Conservation of organizational stability (invariant work rate):** over product lifetime, the average effective activity rate in an E-type program is invariant.
- **Conservation of familiarity:** in order to achieve satisfactory evolution, all the stakeholders should maintain mastery of the content and behavior of an E-type program.
- **Continuing growth:** to maintain user satisfaction over lifetime, the functional content of an E-type program should be continually increased.
- **Declining quality:** the quality of E-type programs appears to be declining, unless they are maintained rigorously and adapted to operational changes.
- **Feedback system:** E-type programs evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base.

6.3.5. Tacit knowledge and implicit information

With the term “tacit knowledge” Polanyi (1967) identifies the phenomenon by which habits and culture that we do not recognize explicitly in ourselves, are used in performing our activities. The tacit knowledge is what end users, as competent practitioners, apply to every task they have to perform. The reasoning and communication among them is supported by the exchange of documents expressed with notations and dialects that depend

on their culture and domain expertise. These dialects and notations represent that information that the users consider important for the achievement of their tasks and reflect the tacit knowledge they share with the other users in the community (Petre, 1995; Green and Petre, 1996; Mussio, 2003).

On the other hand, implicit information phenomenon regards the information that is significant only to those end users who possess the knowledge that is needed in order to interpret it.

6.4. Computer semiotics and HCI

Computer semiotics is defined by Andersen (1992) as

a discipline that analyze computer systems and their context of use under a specific perspective, namely as signs that users interpret to mean something.

Through his work, Andersen defines four different perspectives on sign, through the illustration (Figure 35) of what the author calls map of computer semiotics (Andersen et al., 1993; Andersen, 1997):

- **Signs as system:** the user is considered both creator and user of signs. Creator, in that s/he is the interpreter and referent of signs, and user, in that uses and reproduces code that is the result of a semiotic work done by others.
- **Signs as knowledge:** the user's activities related to learning, and understanding are performed by psychophysiological, psychological and psycholinguistics mechanisms. As to computer-based signs, the mechanisms are studied by cognitive science and ergonomics.
- **Signs as behavior:** the focus is on user's interactions with the environment and on the communication with other users.
- **Signs as artifacts:** the user is considered an innovator of code and an explorer and inventor of signs.

In (Nake and Grabowski, 2001), human-computer interaction is studied as a process of pseudo-communication, in that it shares certain aspects with communicative action, but actually it is not communication. Interaction is seen as a process, that at first sight may appear as a sequence of simple operations related to each other that are taken in turn by two systems, the human and the computer. Nake and Grabowski describe the beginning of an interaction process as the moment in which signs “leave” the human and “enter” the computer becoming instantaneously signals. The signals are computed by the machine and the results of the processing appears on the monitor of the computer. In that moment, the human perceives the signals and transforms them back into signs. The authors highlight how humans can read the signals only as signs and they are embedded immediately into contexts and purposes. In (Nake and Grabowski, 2001), this point of view is framed in a discipline called technical semiotics or semiotic engineering. This discipline constitutes the basis of the approach adopted in this thesis.

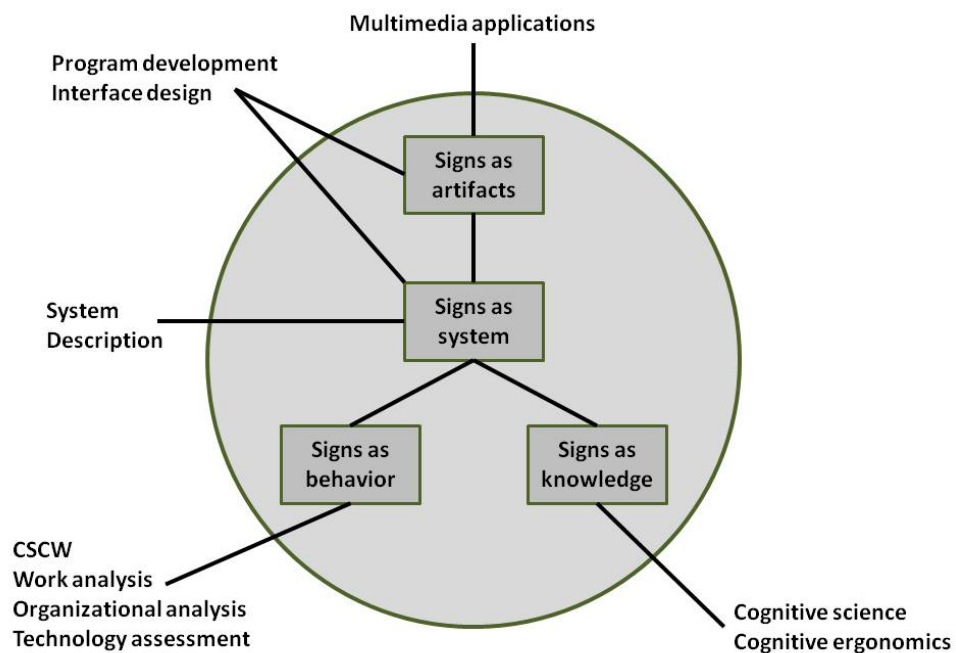


Figure 35. The map of computer semiotics proposed by Andersen (1990).

Semiotic engineering has been proposed by De Souza (1993) and Leitão et al. (2007) as an approach to user interface languages design. It views interactive software systems as metacommunication artifacts. In the process of metacommunication, HCI designers sends

one-shot messages to users explaining how and why they should communicate with the software applications in order to achieve a specific goal.

To decode and interpret the HCI designer's message, the users have to communicate with the message itself. The message gradually unfolds to the users all the meanings that the HCI designer encoded in it. In this view, the message (the software application) serves as the designer's deputy.

In (de Souza, 2005) a four-stage evolutionary schema of metacommunication among designers and users is described:

- The designer studies users, their activities, and their environments.
- The designer expresses her/his view about the users and how they should change together with their activities and environments.
- Users unfold the designer's message through interaction with the software application.
- Users make sense of the designer's message and respond to it.

Going back to the digital communication process model in Figure 27, the whole process can be analyzed by a semiotic point of view and five distinct levels of analysis can be identified (Stamper, 2000; Mussio, 2009; Valtolina, 2010):

- 1) **Physical-empirical level:** regards the management of all the physical, perceivable and transmittable forms of the messages. At this level, the organization of the message is considered separately from its user, from its meanings and from the context in which its meanings are assigned (Figure 36). The correct generation of the pixels on the screen and the correct transmission of signals from a mobile phone to another one are examples of what is considered at this level.
- 2) **Syntactic level:** regards the organization of the message both considering the signals that form it and the set of all possible messages (messages language), independently from the physical and empirical characteristics and from the meaning of the message (see Figure 36). The syntax of the language is considered, in order to check if the message sent respects the rules and to check its validity.

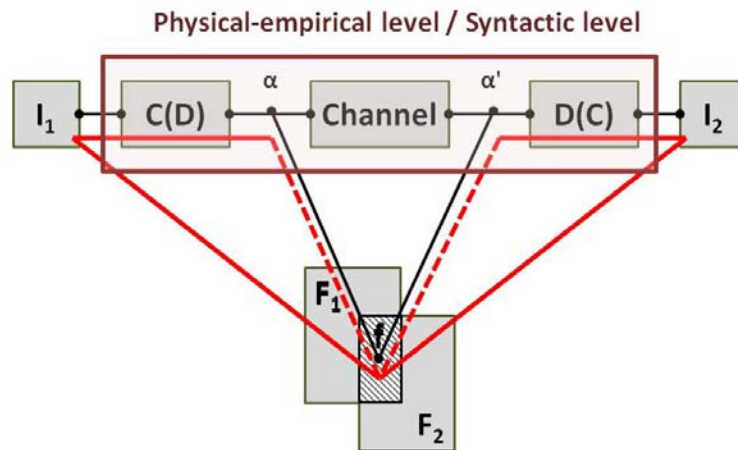


Figure 36. The physical-empirical and the syntactic levels in the digital communication process.

- 3) **Semantic level:** this level studies the meaning and the validity of the meaning that each of the two communicants assign to the message. It extends the two previous levels to the worlds of phenomena, but still the two communicants as individuals are not analyzed (Figure 37).

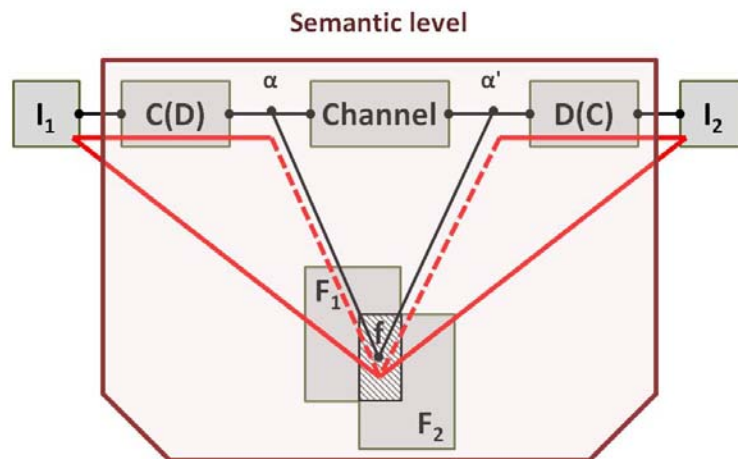


Figure 37. The semantic level in the digital communication process.

- 4) **Pragmatic level:** this level regards both the meaning assigned to a message by the two communicants and the effects that the message produces in the person who interprets it. The two communicants are considered in this case, together with the

other elements of the model considered in the previous levels (see Figure 38). The context in which the communicants operate influences the meaning assignment.

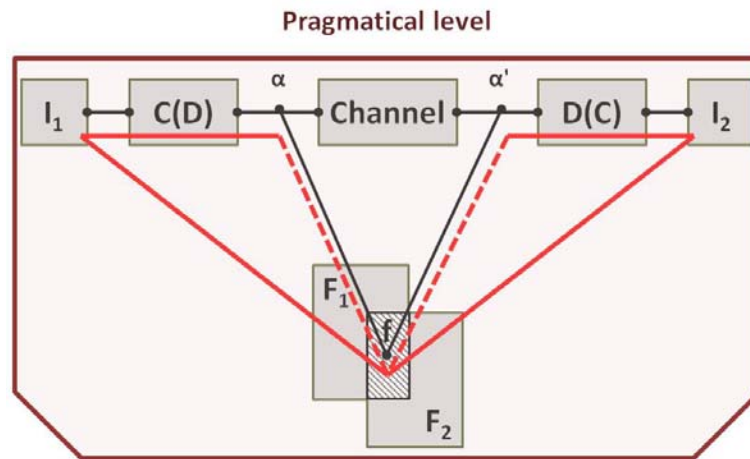


Figure 38. The pragmatic level in the digital communication process.

- 5) **Social level:** regards the effects of the message at a cultural level, considering the relationships among the people involved in the communication (Figure 39). Going back to co-evolution concept, this level observes how the rules and the behavior of users and systems change during the communication process. The community of interest is observed, enlarging the attention to the social behavior of the stakeholders and not considering the interaction process as individualistic but as collectivistic.

The analysis levels are not independent, because each one is based on and influenced by the previous one.

The case documented in (Dale, 2004) is used in what follows to better explain the five semiotic levels in digital communication.

An employee of a company sent a message via an instant-messaging client to his boss. The message was about a 401(k) plan that the employee wanted to suggest to him. Unfortunately, the boss' instant-messaging client rendered the "(k)" string as a pair of red kissing lips.

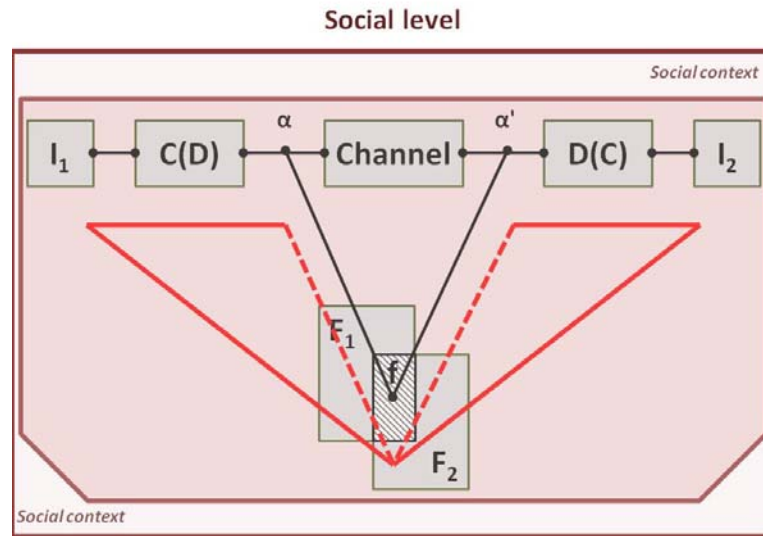


Figure 39. The social level in the digital communication process.

At physical-empirical level, the physical creation and the empirical management of the signals take place correctly, allowing the message sent by the employee to reach his boss.

At syntactic level, rules are applied to check the correctness of the message sent. It is verified if, given the UNICODE definition, the generated codes are correct and if, given the rules for English lexicon, it is verified that the message is composed by correct words. Also at this level, the communication takes place correctly, because from the syntactic point of view the message sent is correct.

At semantic level, meanings are associated to the message. The communicants put in relation what they read with real-world entities and the misunderstanding begins. In fact, the employee associate the string “401(k)” to a retirement plan, while the boss associate the same string to a pair of red kissing lips.

At pragmatic level, the misunderstanding continues in that the two communicants interpret their association according to the reality in which they live. Therefore, the boss interprets the kissing red lips as a message about seduction, while the employee wants to discuss about a retirement plan.

At social level, the choice taken at pragmatic level changes the relationship between the two communicants. This repercussion is an example of the possibilities of influence that

the new media have on the organization of a virtual community and on digital communication.

6.5. The role of the software system in digital communication

Three different types of digital communication may take place:

- person-to-person communication: in this case the communication takes place between two persons in an unstructured way. The digital channel is the medium by which the two communicants exchange their messages and some examples of the tools used for communicating are text processors, e-mails, instant messengers.
- communication through a joint computer-based artifact: in this case the communicants use the same system in order to collaboratively work to reach a common goal. An example of this type of digital communication is the use of ERP systems. The various employees of a company use the ERP system's module dedicated to their sectional activity and the data they save into the shared database is used by the other sectors in the company. In this case, the communication takes place along a channel that allows only structured exchange of messages. However, as highlighted by Robinson (1993), in order to support cooperation among various members of the same organization, a "double level language" is needed: implicit communication, often conveyed by artifacts, and explicit communication are not alternatives but have to be performed in conjunction to better support the collaborative process. Explicit communication may take place by means of annotation tools that allows the users to exchange messages in a way that recalls the previous type of digital communication, the person-to-person one.
- communication between users and developer by means of a program: in this case the interaction of a user with a program represents a communication between the user and the developer of the program, even if the developer is not "active" during the interaction process. In most cases, this type of communication is unidirectional but, as will be shown in the next Chapter, EUD activities transform this communication type in bidirectional. In fact, the end-user developers are able to

extend, modify, or configure the program they use and this activity can be seen as the generation of an utterance sent from the users to the developer.

This classification suggests that the role of a software system in a digital communication process is twofold: it can be seen as an utterance (like in the third type of digital communication in the classification) and/or as a structured channel that supports cooperation among users. As will be discussed in the next Chapter, the software system acts as a mediator among the various communicants who participate in a collaborative process.

In the next chapter of this thesis, the semiotic view on the digital communication process is applied to the context of end-user development in order to describe how the various stakeholders communicate and collaborate. Further on, in Part IV, the architecture for EUD that is presented implements the semiotic view presented in this chapter.

Chapter 7:

A semiotic model for end-user development

This chapter presents the semiotic model for EUD that is an evolution of the semiotic view on digital communication process described in the previous chapter. This semiotic model describes how the end users participate in the design process and collaborate together with the other stakeholders to achieve a common goal. In particular, the role of the interactive software system as mediator in the communication among all the stakeholders is formalized.

7.1. A semiotic view on end-user development

As illustrated in the previous chapter, the semiotic engineering sees the interactive software systems as metacommunication artifact. The software systems are one-shot messages sent by the designer to the user, that should explain the user how to use the system in order to reach a specific goal. They are one-shot messages in that they convey a complete content that is encoded in them and that is made available by the system's interface. The metacommunication is fully achieved if the user communicates with the message, i.e. if the content created by the designer with a specific intent is interpreted as such by the user.

In an end-user development context, the roles of end user and designer partially overlap. A particular scenario is depicted in Figure 40.

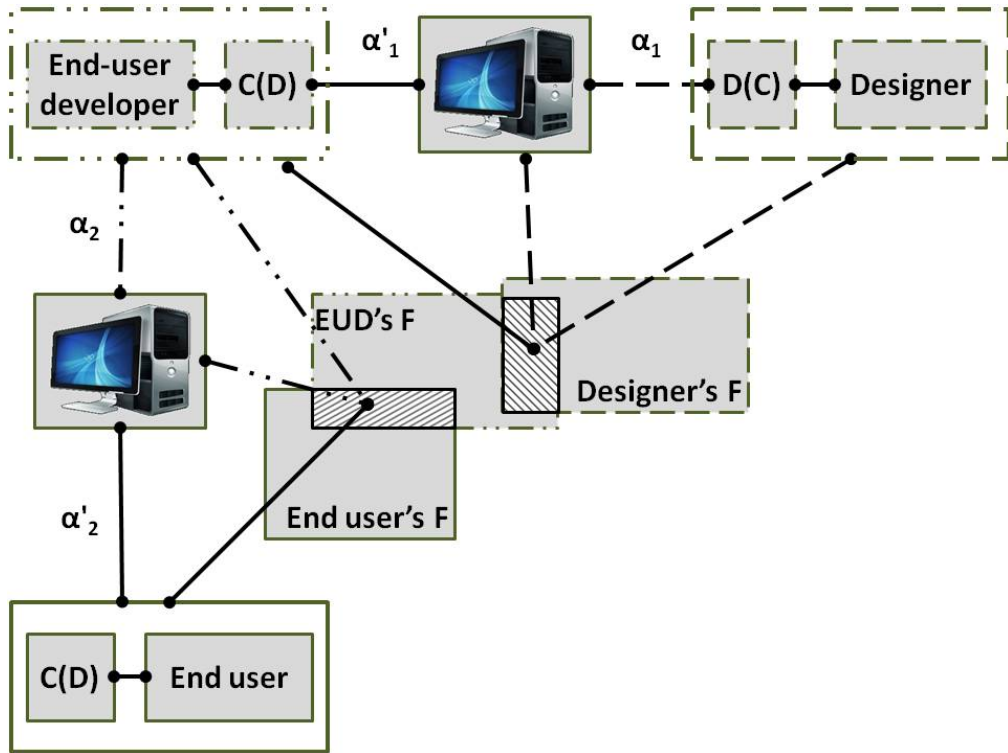


Figure 40. The metacommunication process between the designer and the end-user developer is illustrated in the top of the picture. The metacommunication between the end-user developer and the end user is depicted in the left part of the picture.

On the top of the picture, the metacommunication between a designer and an end-user developer is depicted. The metacommunication takes place by means of the interactive software system (the message) α_1 . The system acts as deputy of the designer and represents a one-shot message for the end-user developer. At this stage, the end-user developer acts as an end user, by using the system that the designer developed for her/him. However, by the use of the system, the end-user developer begins to develop a new system for another user, called simply end user in the picture. This second metacommunication is performed by means of the system developed by the end-user developer, α_2 , that is the one-shot message sent to the end user.

This scenario shows how several worlds of phenomena have to be combined in a context in which end-user development is used as approach to support the collaboration in design. In

particular, in a context in which the communities involved in the collaboration are global, as discussed in Chapter 2, the achievement of a mutual understanding among all the stakeholders requires several cycles of metacommunication. In this scenario, the metacommunication can be seen as a tool to support a progressive collaborative semantization process.

Another possible scenario is the one in which the end-user developer develops a system for her/himself and not for another end user. This case is illustrated in Figure 41. In this case, the end-user developer plays both the roles of designer and user of the system and the metacommunication takes place with the presence of both the figures. Since the sender and the receiver of the message are the same person, unlike the previous picture (Figure 40), the message sent by the end-user developer and received by the end user remains the same.

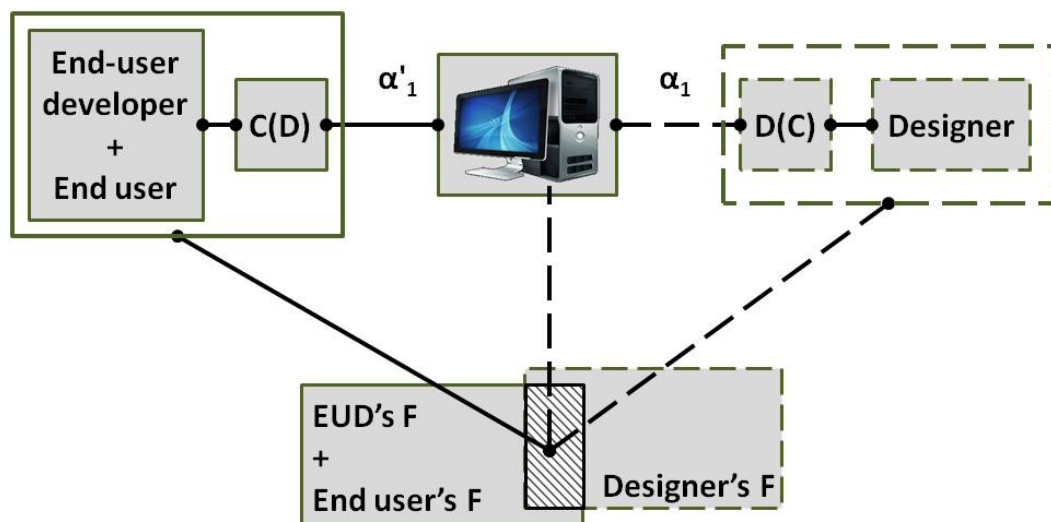


Figure 41. Another scenario of metacommunication where the end user developer develops a system for her/himself and becomes therefore designer and user of the new system.

7.2. The interactive software system as a mediator

In the metacommunication process between the designer and the end user (or end user developer – as depicted in Figure 41), the interactive software system plays the role of the mediator. This situation is in fact described by the third type of digital communication described in Section 6.5 of the previous chapter.

By definition (Boulle, 2005), a mediation process allows two human actors (shortly actors) to reach a common understanding, related to a specific domain, by the support of an agent, the Mediator. In a collaborative design context, the mediation process consists of exchanging messages between the two actors playing a certain role in the collaboration. Such actors are generally members of different communities of practice (Wenger, 1998) that constitute a community of interest. As previously illustrated in this thesis, a CoI is defined as a group of people who share a concern and have to reach a common goal on something they have to collaboratively perform.

However, communication is just one of the challenges that emerged from the research context described in Part I and in particular in Chapter 4. In fact, also system customization to users profile and co-evolution of users and systems should be considered.

As to system customization, each stakeholder should be provided with an interactive software system (i.e. a virtual environment) localized to her/his language, notation and culture, to the role s/he plays in the CoP, and to the digital platform s/he uses. This kind of environment should support the stakeholders in the communication process with each other, allowing them to create and exchange objects as tools of reasoning. These objects are called boundary objects, that is artifacts that collaborating stakeholders use in order to reach a common understanding about the activities they are performing. Boundary objects provide a means for team members to interact, react and negotiate around a concept using concrete representations to create a common language and experience for dialogue and critique. In the semiotic model used for this research, the boundary objects are used in the collaborative process and are enriched with annotations.

As to co-evolution, the system life cycle starts with the design and implementation of a first version of the system, which is, thereafter, used by end users on the field. End users not only use the system, but also modify, extend, and configure it with EUD activities. The end users improve their knowledge of the system, update their way of proceeding in the real world, and even modify their work organization. Therefore, EUD activities offered by the system, supports the evolution of the user and of the system, facing the co-evolution challenge.

The interactive software system, together with the boundary objects and the annotation tools constitute the mediator in the metacommunication between the different stakeholders involved in the collaboration.

In this view, the digital communication that takes place between the user and the developer by means of the software system becomes bidirectional because the end users who perform EUD activities are able to extend, modify, or configure the software systems they use and this can be seen as the generation of utterances sent from the users to the developer.

7.3. Mediation process and mediation mechanisms

All the environments provided to the CoPs' members, should be equipped with agents devoted to the translation of the exchanged boundary objects and annotations, i.e. the messages. In this case, the translation is the process by which the boundary objects and the annotations are adapted to the languages and domain specific notations of the members.

The first two steps of a generic mediation process are illustrated in Figure 42.

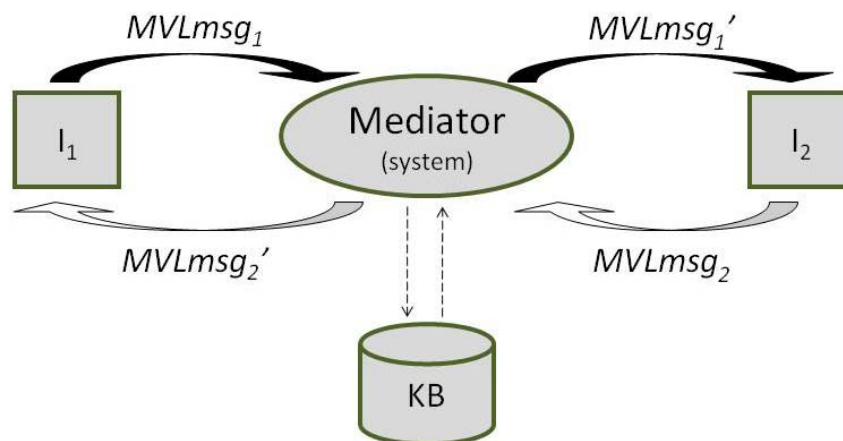


Figure 42. The first two steps of a generic mediation process between an end user and an end-user developer.

The first step, represented by the black arrows at the top, starts with I_1 , a human actor, who sends a message ($MVLmsg_1$) to another human actor I_2 . Before reaching I_2 , $MVLmsg_1$ is

captured and managed by the Mediator that, by exploiting the knowledge base (KB) for the current domain, translates it into the interaction language used by I_2 , so that s/he can understand it. The translated message, $MVLmsg_1'$, is then delivered to I_2 .

The second step of the mediation process is represented in Figure 42 by the white arrows at the bottom. I_2 replies to I_1 's message by sending a new message ($MVLmsg_2$) to I_1 . Also this message is captured and managed by the Mediator that, by exploiting the knowledge base, translates it into the interaction language used by I_1 . The translated message ($MVLmsg_2'$) is then delivered to I_1 .

The elements involved in a mediation process constitute a Mediation Mechanism (MM), defined as:

$$MM = (\text{Mediator}, \text{KB}, \text{MVL})$$

where:

- **Mediator** is the agent that supports the two human actors in reaching an agreement through a mediation process;
- **KB** is the shared knowledge base related to a specific domain in which the actors collaborate;
- **MVL** (Mediation Visual Language) is the visual language constituted by the set of messages exchanged by the two human actors by means of the Mediator, and is defined as follows:

$$MVL = \{MVLmsg_1, \dots, MVLmsg_n\}$$

A **MVLmsg** is a message defined as:

$$\text{MVLmsg}_i = \langle \text{data}, \text{metadata} \rangle$$

Data describes the object of the mediation:

$$\text{data} = \langle \text{EP}, \text{A} \rangle$$

where:

- **EP** is an executable program, that is the software system that the two actors are collaboratively designing or part of it
- **A** is the annotation that the sender attached to EP in order to communicate with the receiver

Metadata specify some characteristics of the sending and receiving actors and of the digital platform used:

$$\text{metadata} = \langle \text{S}, \text{R}, \text{PI} \rangle$$

where:

- **S** is the profile of the human actor that acts as a sender
- **R** is the profile of the human actor that acts as a receiver
- **PI** is the specification of the hw/sw platform being used

When an actor sends the first MVLmsg, a mediation process starts. All the messages following the first one convey the contributions of the actors involved in the

communication. The metadata related to the profiles of the communicants are sent just in the first MVLmsg and not repeated in the successively messages.

7.4. The Software Shaping Workshop methodology

As illustrated in Chapter 2, in literature several method and tools to support the participation of the end users in design and in particular the end-user development approach.

In this research, the Software Shaping Workshop methodology (SSW) has been used (Costabile et al., 2007b). SSW allows to design and develop virtual environments that:

- support the activities of users acting a specific role in their community and having a specific application domain;
- are tailorable, customizable and adaptive to the working context;
- support the exchange of information among users belonging to different communities;
- are multimodal and interactive.

The methodology is evolutionary and participatory: the final user can customize and evolve her/his own virtual environment and s/he is involved in each step of the system development.

All the phenomena that affect the digital communication process, listed in Chapter 6, are faced by SSW: user diversity, tool grain, communication gap, implicit information and tacit knowledge, and co-evolution.

Users belonging to different communities usually are accustomed to different language, culture, skills, abilities (physical and cognitive), preferences and role. In a working context, the communities' members cooperate performing different tasks and exchanging documents that are part of a shared knowledge base. The correct comprehension of a document depends on the user's application domain and on his/her language and specific

dialect (Fogli et al., 2005); therefore, tools are needed, which are customized to user's needs in order to allow communication and comprehension among the different communities they belong to. SSW supports the design and development of internationalized systems localizable to the culture and skills of the target community of practitioners and tailored, customized and adapted to the context of activity.

SSW methodology considers the development of two different kinds of environments (called workshops):

- The **application workshop** is a virtual environment customized to each member of the community, according to her/his performing task and role, to her/his culture and language, and to the digital platform in use.
- The **system workshop** is a virtual environment that permits to customize the application workshop to users' preferences, characteristics and needs.

With this idea in mind, workshops are organized into a three-level network (Figure 43), in which each member of the design team (software engineers, HCI experts and domain experts) collaborate to design and develop virtual environments customized and tailored for their activity domain and performing tasks:

- at the top, **meta-design level**, software engineers use a system workshop to create other system workshops in order to permit other software engineers, HCI experts and domain experts to collaborate to design and development of application workshops;
- at the middle, **design level**, representatives of the end users (domain experts) and HCI experts collaborate, using their own system workshops, for designing and implementing application workshops; the domain experts are end-user developers that are in charge of designing and developing systems to be used by other end users.
- at the bottom, **use level**, end users use the application workshops designed and developed at design level in order to perform their task.

The arrows in Figure 43 represent the ways in which the communication among the various communities of practice is guaranteed. Dashed arrows indicate the paths of communication that exist among the CoPs that work at the same level in the SSW network. As an example, at use level, data related to the activity at hand is exchanged, while at design level, HCI experts and domain experts exchange the results of their design in order to collaborate in developing the application workshops. The solid arrows indicate the way in which lower levels communicate with the higher ones and vice versa. This communication is supported by the use of annotation tools that allow the users to annotate their problems and to send their comments to the experts that are available in the SSW network.

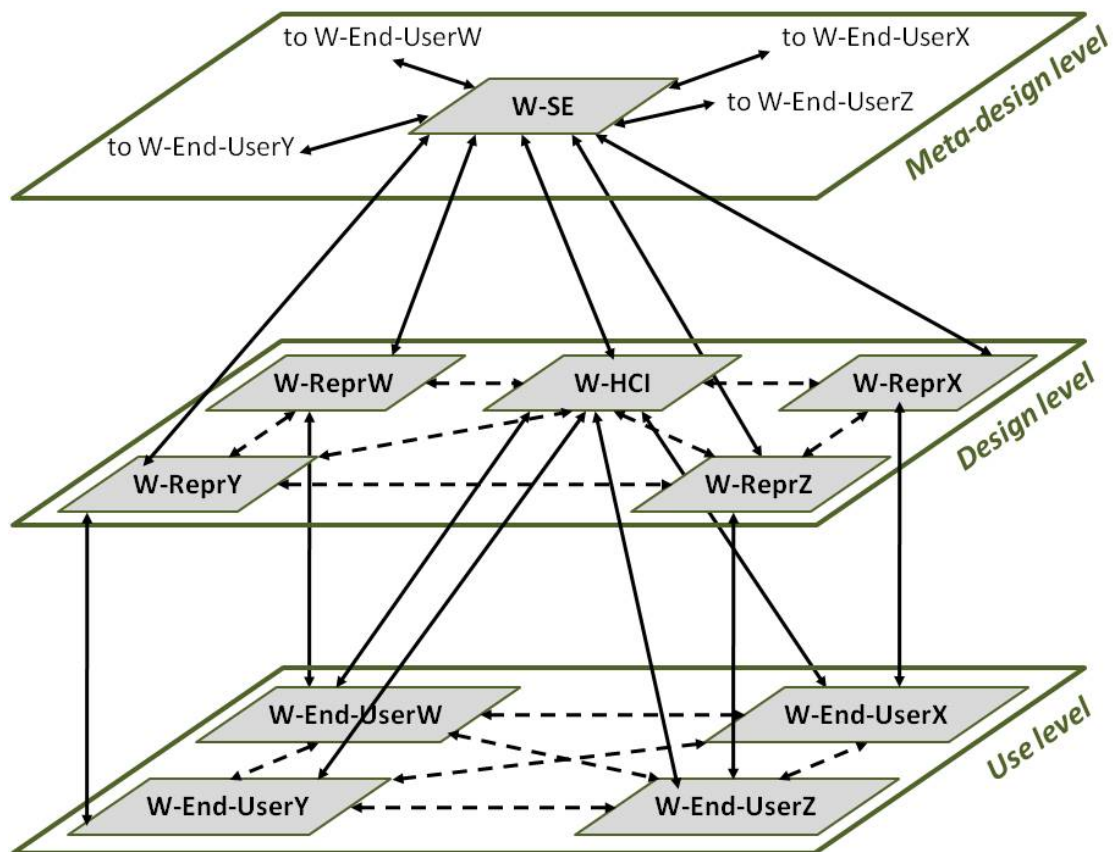


Figure 43. The SSW network. The arrows describe the communication processes that take place among the various stakeholders working at the different levels (Costabile et al., 2007b).

The different levels in the SSW network are represented in Figure 43 as flat layers each of them hosting various communities of practice who collaborate “on the same level”.

However, in the application of SSW methodology to real cases, the levels may become three-dimensional layers in which the CoPs involved in the project operate at different sub-levels into the same level. An example is given in the factory automation case study presented in details in Chapter 9.

Each domain expert is a stakeholder that evaluates the system considering it from her/his perspective biased by her/his different cultural backgrounds, experiences and standpoints of problems. Thus, a communication gap arises among the component of the design team: software engineers, HCI and domain experts adopt different approaches to abstraction and follow different reasoning strategies to model, perform and document the tasks to be carried out in a given application domain; furthermore, each expert expresses and describes such tasks adopting her/his own notation and dialect (Fogli et al., 2005).

Communication among the application and the system workshops is supported by the annotation tool. At the use level, end users exchange data related to their current task in order to cooperate in achieving a common goal. At the design level, HCI experts and domain experts exchange programs specifying the workshops they are going to develop. HCI and domain experts also communicate with software engineers when it is necessary to develop new tools for supporting their activities. The lower levels are connected to the upper ones by communication paths, allowing end users and designers to interact with other workshops annotating their problems and communicating them to all the experts working in the same SSW network (Costabile et al., 2007b).

In the SSW methodology, the principle of symmetry of ignorance is accepted: the software engineers are owners of the technology, the HCI experts are owners of the usability methods and the domain experts are owners of the domain problems. A collaborative approach to development is adopted permitting the members of the design team to cooperate: the clash of different cultures that normally determines communication gaps among the design team members is avoided thanks to the localization of each system workshop that supports the stakeholders' activity. That permits the workshops to represent spaces and places that serve as boundary objects where different cultures can meet (Fischer, 2000). As described in Fogli et al. (2005), the SSW methodology is multi-faced

because it permits each stakeholder involved in the design process to cope with problems of the system observing them from the perspective of their own culture.

In the next part of this thesis, Part IV, the architecture for EUD is presented. It implements the semiotic model described in this chapter, allowing the participation of the end users in the collaborative design process and supporting the communication among all the stakeholders involved.

**Part IV:
An architecture for
end-user
development**

Chapter 8: BANCO architecture

This chapter presents an architecture, BANCO architecture, for end-user development supporting global communities. The architecture implements the semiotic model for EUD presented in the previous part of the thesis. The architecture is the result of a process determined by the experiences performed in several case studies (illustrated in Part II) in which different communities of practice needed to collaborate and perform end-user development activities.

8.1. B-architecture for EUD supporting global communities

This chapter presents an architecture, the BANCO (Browsing Adaptive Network for Changing user Operativity) architecture for EUD supporting global communities. This architecture allows the implementation of the hierarchy of SSW, it is Ajax-like, component-based and Web service-based. It supports collaborative activities, among users belonging to different communities and cultures, and allows the exchange of multimedia documents and annotations supporting the generation of shared knowledge bases. BANCO architecture (B-architecture for short) is developed following an approach that underpins re-use and evolution of software.

This approach respects the rules of Web 2.0 (O'Reilly, 2006), in fact software artifacts developed on the basis of the B-architecture:

- a) are developed as incremental prototypes, in that at each step new features are added to them (component-based development);
- b) are considered perpetual beta, in that they are no more considered as products but as processes of engagement that involve also the end users;
- c) are developed permitting to share data and services for being reused by others (Web service-based development);
- d) reside in space between devices, i.e. they are not residing only on clients or on servers (Ajax-like).

As illustrated in Chapter 4, three main challenges emerge from the research context described in the first three chapters of this thesis, and regard the necessity of supporting system customization, system evolution, and communication.

8.2. B-architecture and system customization

System customization challenge refers to the problems arising from the research context and described in Chapter 3. Internationalization and localization techniques represent the response to the issues that affect end-user development activities performed by the members of global communities.

The necessity of design and develop software systems that are adaptable to the different cultural characteristics of the stakeholders in a collaborative project, leads to the definition of three main aspects to be considered in the internationalization and localization process: users' culture, users' role in the context, digital platform in use.

In localizing the systems, the culture of the user is taken into account. For culture both the language and the cultural conventions deriving from the country of origin are considered.

This means that not only a translation activity is required, but also an adaptation of colors, shapes, and topology of the graphic elements appearing on the screen is needed.

B-architecture uses specific languages that allow to describe a software system by distinguishing its contents organization, its contents localization, and its materialization of digital platforms.

The GILT methodological model is applied and the architecture follows the third method of GILT implementation described in section 3.4.4: the system is developed dividing what is the “core” of the application (neutral in respect to the locales considered) by the localization files that maintain all the values for the parameters that are affected by cultural characteristics.

As introduced in the conclusions of Chapter 3, B-architecture implements what was proposed by Yeo in (Yeo, 1996): he proposed the concepts of “cultural user interface” and “personal user interface” and considers cultural aspects not only the language spoken in the country of the user and the related conventions but also the expertise possessed by the user and her/his background.

Localization according to the role that the users play in a specific context is in fact one of the three aspects to which the systems realized on the basis of the B-architecture are localized. The role of a user in a community influences the choice of tools and the notations to be used in the software interface in order to make it usable for her/him. In each of the cases presented in Chapter 5, various domain experts were collaborating to reach a common goal by serving the community offering their expertise.

As to users’ culture, B-architecture is internationalized in respect to the colors, the shapes and the topology of the entities that constitute the systems’ interface. According to what illustrated in Chapter 3, colors are interpreted differently in different cultures and play a fundamental role in interface design because of the information that they convey. The same importance has to be given to the shapes of the entities that are designed because symbols and images convey meanings as well and a wrong choice could compromise the metacommunication from the designer to the users. Topology in the sense of the study of space, dimensions and positions of elements in an interface is affected by cultural

conventions as well. In fact, as discussed in Chapter 3, the writing system of a language and the correspondent reading/writing direction affects the way in which information are recognized by the users in an interface.

The third localization type considered by B-architecture is the one for the digital platform in use. The same software system could be accessed and used by users using different digital platforms and device, having different display dimensions and resolutions and offering different pointing device. According to these characteristics, the system has to be adapted in terms of resolution of the images, organization of the interface and giving priorities to the elements to be materialized on screens.

The implementation of this threefold approach to localization in the B-architecture is described further in this chapter.

8.3. B-architecture and system evolution

End-user development is one of the disciplines in which the co-evolution phenomenon becomes more evident.

The evolution of the users during their interaction with the system forces the system designers to adapt it to the changes that occur (Nielsen, 1993; Bourguin et al., 2001; Carroll and Rosson, 1992; Arondi et al., 2002; Costabile et al., 2006a).

The process of design and development becomes therefore a never-ending activity, in which the separation between design and use time is blurred. The “design-in-use” continuum allows the creation of continuously evolvable systems, extendable with the collaboration of all the stakeholders involved.

In this perspective, B-architecture is following the perpetual-beta approach (Bruns, 2008), considering the software systems no more as products but as processes of engagement. Moreover, supporting EUD activities, B-architecture faces the co-evolution challenge involving actively the end users in the design and development process. The end users

become end-user developers and are therefore able to deal by themselves with the adaptation of their systems in order to fulfill their needs that evolve in time.

B-architecture supports EUD activities by allowing users to reuse software components previously realized by others. According to the definition of end users that is adopted in this thesis and is highlighted in Part I, the end users are not computer science experts and/or professional developers. Therefore, B-architecture allows to develop workshops dedicated to end-user developers in which they are allowed to compose their systems without writing any code but just choosing existing entities. The end users will see those entities not as software components but as visual entities with specific behaviors and will be able to compose their system by means of direct manipulation interaction style.

8.4. B-architecture and communication

The communication challenge is faced by the B-architecture by offering annotation tools.

Annotation is a typical tool used to add information when working on a document: an annotation can be a note for the single user or a note to be shared and discussed with other persons to exchange and accumulate knowledge (Marcante and Mussio, 2006). An annotation is referred to a part of a document: the document is called target document and the annotated part of the document is called the base of the annotation. The evolution of Web 2.0 technologies leads to a fuzzy distinction between documents creators and users, and opens to new possibilities of collaboration and work organization. The nature of documents has changed in time in six phases:

- 1) the traditional paper-based document written by a single author, annotable by her/him or by other persons and physically exchangeable;
- 2) the e-document modifiable only by its author, usable by other users, exchangeable by e-mail or other digital communication channels;
- 3) multimodal (audio and video) e-document modifiable only by its author, usable by other users;

- 4) annotable e-document created by a single author and annotable remotely by other users – the annotations are available to all the document's users that become therefore producers and not only consumers of the document and of the annotations;
- 5) multimodal annotable e-document created by a single author, materialized on different channels and annotable by all the users (who become producers and not only consumers).
- 6) multimodal annotable e-document created collaboratively by a group of authors, materialized on different channels and annotable by all the users (who become producers and not only passive consumers).

B-architecture annotation tools allow the implementation of interactive systems as documents of the sixth type: e-documents that are created in a collaborative way by many stakeholders involved in a common project.

With annotation all the stakeholders involved in the collaborative work are able to report about usability problems they find in interacting with the system they are using or to communicate to others ideas and requests they may have about the ongoing project.

A note is a comment that is associated to a document (called target document). To a first note, further notes can be added, not necessarily created by the same author of the first one.

The set of notes belonging to the same discourse constitute a thread, defined as

$$\text{Thread} = \{\text{note}_1, \dots, \text{note}_n\}$$

where n is an integer number.

The notes belonging to the same thread are ordered on the base of their creation date and time.

On the target document, the presence of one or more notes is signaled through the materialization of a visual link (vl). The visual link is an active button that allows to open the thread of notes associated to it.

An annotation is therefore defined as:

$$\text{Annotation} = \langle \text{Thread}, \text{vl} \rangle$$

Each note is composed by:

- A reference to the target document to which the note is related.
- A title chosen by the author of the note.
- The body of the note (the text that was typed by the author of the note).
- The author's name.
- The coordinates to the point of the target document where the note is attached.
- Date and time of creation.
- If needed, a reference to one or more multimedia resource associated to the note.

In order to allow the retrieval of annotations and documents, each time that a note is created, a set of three indexes are generated. The three indexes are called: automatic, extracted, and personal indexes.

The automatic indexes are the ones that are associated to the note automatically by the system:

- Target document.
- Date and time of creation.
- Author of the note.
- Title of the note.

The extracted indexes are the ones obtained by the use of a domain-related glossary: the system parses the body of the annotation looking for terms that are included in the glossary

and considers those terms as indexes for the note. The user is also allowed to insert new terms in the glossary so to add more knowledge to the existing archive.

The personal indexes are the indexes chosen directly by the user. Those indexes are terms that the user considers relevant for the note at hand and that s/he does not want to add to the glossary for future use.

The construction and maintenance of shared knowledge bases supports the collaboration in communities of interests where different communities of practice work together to common projects. Shared knowledge bases represent a source of information and inspiration to be used for the current projects but also for the future ones and could be a resource for quickly solving problems that emerge in multidisciplinary design.

8.5. B-architecture overview

An overview of the B-architecture structure and of its elements is given in Figure 44. Client and server side of B-architecture are described separately in the following subsections.

8.5.1. B-architecture client side

On the client side, the B-architecture is constituted by four elements: BANCO Engine (BE), the specification of the initial state and the dynamics of the application (SPEC), the Current State Description (CSD) and the I/O Manager.

BANCO engine (BE) is an application-independent interpreter, i.e. it is used for each instance of application created on the B-architecture because even if the application domain changes BE is neutral respect to it. It manages the instantiation of the localized application and the interaction process through the interpretation of the specification documents (SPEC).

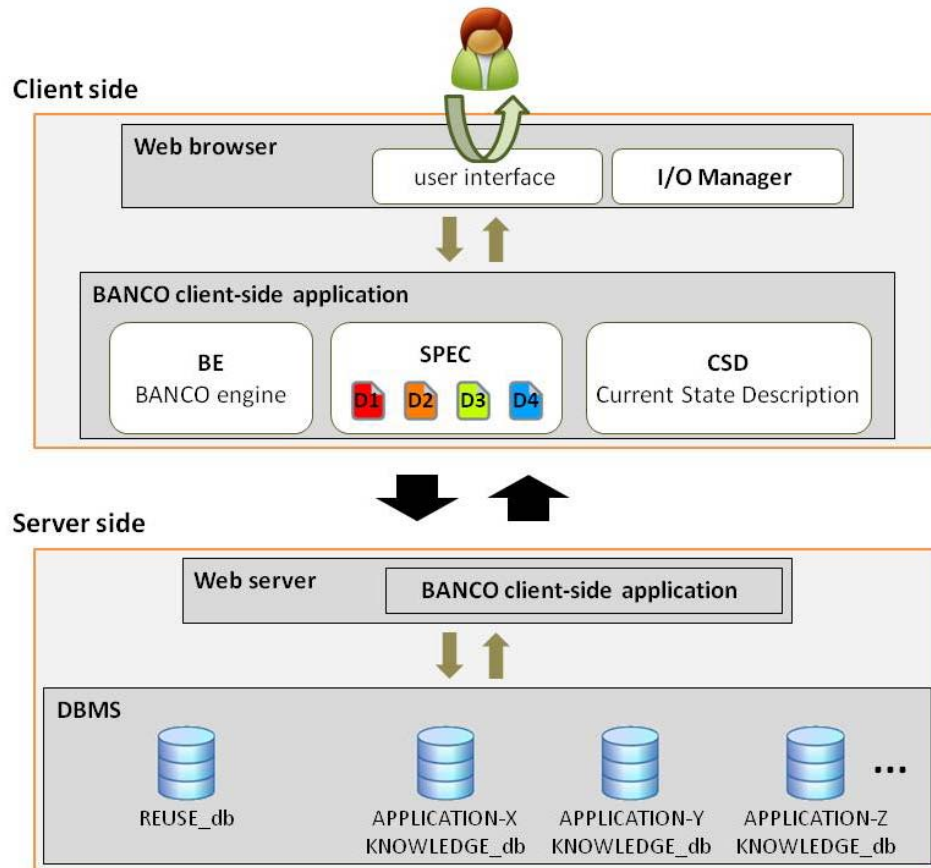


Figure 44. An overview of BANCO architecture.

The SPEC specification, is given by four documents written in different languages to be interpreted by BE. The four documents specify:

- D1: the description of the initial states of each entity that constitutes the current instance of the application abstracting from localization and materialization. For example, a type of entity “operator” represents an element associated to the execution of a computational activity. At materialization time, this entity “operator” is instantiated as a button or a menu item according to the context of use that is for example the mechanical engineering context.
- D2: the localization parameters of the current instance of the application. In the “operator” example, the entity is defined according to Italian. The Italian culture states that the label to be put on the entity will be in Italian language, that the label will be “Avvia”, and that the writing/reading direction for Italian language is from

left to right and from the top to the bottom. Moreover, the technical standards related to the context of use suggest that the color of selected state representation should be green, that the entity should have black borders, and that the entity should be colored in light gray in the non-selected state.

- D3: the materialization parameters of the current instance of the application. As to the “operator” example, in D3 it is stated that the button will have a rectangular shape of a certain size and its borders will have a given thickness.
- D4: the dynamics of the current instance of the application. In this document, the computational activity associated to the “operator” is described.

CSD, the current state description, is created by the BE by interpreting the SPEC documents. BE creates the initial state of the environment by using a library of instantiation functions, and then manages the dynamic of the environment (environment’s reactions to the user’s actions), by using a library of interaction management functions. The presence of an I/O manager allows the dynamical creation of the current configuration and the management of the I/O interaction between the user and the environment. These are made possible by communicating with the BE in different steps of the session.

8.5.2. B-architecture server side

On the server side, the B-architecture is constituted by two elements: an archive of existing software and configuration components (REUSE_db) and the knowledge base related to the current application (APPLICATION-X KNOWLEDGE_db).

The REUSE_db archive is accessible by the whole community of BANCO users and developers. In it all the documents describing the specification of the initial state of all the possible virtual entities are stored. The documents are used both to materialize the system in use by the user and to be used for the design and development of new systems. In this archive, also copies of ongoing projects may be stored in order to make them available the whole community. This element can be seen as the blackboard for the people who perform end-user development activities using the B-architecture.

The APPLICATION-X KNOWLEDGE_db is accessible by the community of the developers of the specific application. In it all the annotations created by the users and related to a specific application “X” are stored. Also the indexes that are assigned to the annotations are stored in this archive.

8.6. B-architecture implementation

In this section, the implementation of each element constituting the B-architecture, both on client and on server side, is described.

8.6.1. BANCO engine and specification documents

At the base of the current implementation of the B-architecture there is a software engine, called BANCO engine, that specifies the behavior of the software systems. BANCO engine is composed by two libraries of scripts: a) the client-side application, constituted by a set of scripts in ECMAScript language (ECMAScript); b) the server-side application, constituted by a library of scripts in PHP language (PHP).

On the client side BANCO Engine given the Specification documents and the Current State Description, performs the materialization of the initial state of the system through the I/O Manager and manages the interaction of the user with it. As illustrated before, four distinct documents define the specification of a system developed on the B-architecture. For each document, a specific language is used. The languages used in the current implementation of the architecture have been defined in (Fogli et al., 2004; Fogli et al., 2006; Barricelli et al., 2009b; d’Amato, 2009). A work on revision of the languages definition is part of the ongoing PhD project developed by Claudia Iacob at Università degli Studi di Milano (Iacob, 2010).

Document D1 describes the initial states of each entity that is materialized on the screen. D1 is written in IM²L (Interaction Multimodal Mark-up Language) language and describes the virtual entities abstracting from localization and materialization aspects. The schema of IM²L language is depicted in Figure 45.

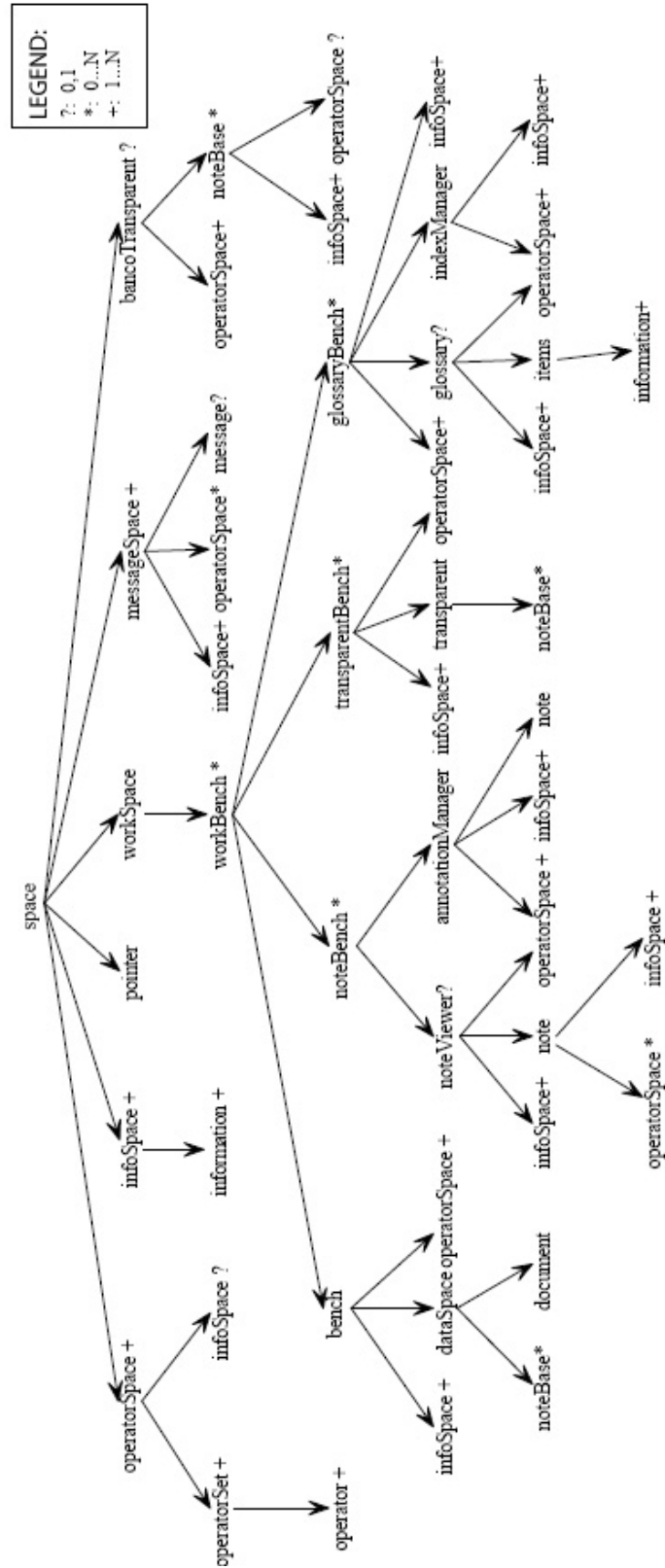


Figure 45. The schema for the IM²L language.

Each node in the schema defines a type of entity which may appear in a software system developed on the basis of the B-architecture. For example, a type of entity “operator” is associated to the execution of a computational activity (examples are buttons, menu items, etc). An “operator” entity belongs (is superposed in the interface) to an “operatorSet”, which can be composed by only one “operator” or many instances of it (for example, a menu item belongs to a menu, while a button belongs to a button panel).

Therefore, the description given by document D1 regards the organization of the interface of the system in terms of elements and relations among them. Document D1 is defined for a specific application domain and a specific role played by the user in a context.

The IM²L description of the entity “operator” is given in Figure 46. In the example, the operator is part of an entity “operatorSet” that has only one element in it. The operator is a button of type “close”, it is materialized in a “non-selected” state and if clicked the associated computational activity “close.js” is executed.

```
...
<operatorSet id="ve11-x" type="nav-tool" name="Navigation tool area" ordered="yes"
state="0">
  <operator id="ve111-x" type="close" name="close button" state="non-selected"
compute="click:close.js" />
</operatorSet>
...
```

Figure 46. An example of “operator” definition given in document D1.

D2 is written in LML (Localization Markup Language) language and gives all the values to be assigned to the parameters that affect the materialization of the system interface in terms of localization. In fact, all the information regarding colors to be used, texts translations, writing direction, cultural conventions are specified in this document. For the “operator” example, the LML code is given in Figure 47.

On the other hand, all the values to be assigned to the parameters that affect the materialization of the system interface in term of adaptation to the digital platform, are given in document D3, that is written in TL (Template Language) language. The “operator” definition in TL language is presented in Figure 48.

```

...
<colors>
<color id-ref="ve111-x" state="non-selected">light grey</color>
<color id-ref="ve111-x" state="selected">green</color>
...
</colors>
<text_direction="rtl" unicode-bidi="bidi-override" />
<space_exploration="ttb" />
<content>
<label id-ref="ve111-x">
<![CDATA[Avvia]]>
</label>
...
</content>
...

```

Figure 47. An example of localization definition for an “operator” given in document D2.

The dynamic of the system is specified in a fourth document, D4, that in the current implementation is written in ECMAScript language. D4 gives the specification of the behavior of the system for a specific application domain.

```

...
<shapes>
<shape id-ref="ve111-x">
<rect width="40" height="60" stroke-width="2" />
</shape>
...

```

Figure 48. The example of D3 document for the “operator” entity.

The I/O Manager is the component of the Web browser that is in charge of managing the visual materialization of the interface of the system. The language of materialization is defined by the information given in the D3 document. For HTML (HTML) materialization, the I/O Manager is embedded in all browsers (the ones that are not textual only). For the materialization in other languages, for example in SVG (SVG), third-part plugins are needed and constitute the I/O Managers.

8.6.2. BANCO knowledge archives

On the server side BANCO engine allows to access both the REUSE_db and the APPLICATION-X KNOWLEDGE_db archives.

Since all the specification documents, annotations and indexes are written in XML-based languages, the server-side archives management is based on the use of a XML-native database, eXist (eXist). This choice facilitates the interoperability among machines and the independence from the different digital platforms.

```
<?xml version="1.0" standalone="no"?>
<rdf:RDF xmlns:a="http://www.w3.org/2000/10/annotation-ns#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <a:Annotation>
    <a:annotates rdf:resource="Valchiavenna.jpg"/>
    <a:title>Villa di Chiavenna</a:title>
    <a:body>Magnifici i percorsi naturalistici!</a:body>
    <a:author>Francesco</a:author>
    <a:context>198 263</a:context>
    <a:created>2010-08-21T18:41:09</a:created>
  </a:Annotation>

  <a:Annotation>
    <a:annotates rdf:resource="Valchiavenna.jpg"/>
    <a:title>Sentiero della Dogana</a:title>
    <a:body>Da percorrere ma difficile</a:body>
    <a:author>Andrea</a:author>
    <a:context>198 263</a:context>
    <a:created>2010-08-22T15:23:45</a:created>
  </a:Annotation>

</rdf:RDF>
```

Figure 49. An example of RDF document for a thread of notes that constitutes an annotation. Two distinct notes are present and are created by two different authors in two different times but referring to the same point of interest on the map of Valchiavenna region.

As illustrated in (Chaudhri et al., 2003), eXist is an open-source XML-native database system that covers most of the basic native XML database features and also many advanced techniques like keyword searches on text, queries on the proximity of terms, and

regular expression-based search patterns. eXist most important point is the index-based query processing, in fact its search engine has been designed to provide fast XPath (XPath) queries by using indexes for all element, text and attribute nodes. A wide range of query expressions are processed on the basis of their index only, so that eXist does not load the nodes unless it is required, for example for query results displaying.

The annotations are stored in files written in RDF language and using the Annotea Annotation Schema (Annotea Project). An example of annotation is depicted in Figure 49.

For each note in a thread of annotation, a RDF file is created and all the three types of indexes described in Section 8.4 are written in it. The RDF documents, for both annotations and indexes, are stored in the APPLICATION-X KNOWLEDGE_db archive and constitute part of the knowledge base of a specific domain.

8.7. B-architecture at run time

When a user accesses a system, the Web Browser loads the BANCO engine client-side application. BANCO engine starts the instantiation of the system by performing the following actions:

- It checks the user profile (in terms of culture, role, and digital platform in use).
- It loads the IM²L specification of the system loading it from the REUSE_db archive (involving therefore the BANCO engine server-side application).
- It loads the LML specification of the system fitting the user profile from the REUSE_db archive (involving therefore the BANCO engine server-side application).
- It loads the TL specification of the system associated to the application from the REUSE_db archive (involving therefore the BANCO engine server-side application).
- It loads the specification of the dynamics of the system from the REUSE_db archive.

- It loads all the existing knowledge (annotations) related to the system to be instantiated from the APPLICATION-X KNOWLEDGE_db archive.
- It materializes through the I/O Manager, the initial state of the system.

The state of the system materialized in a Web browser in a certain instant of time, is what is called Current State Description of the system and is given by the DOM of the current application.

Whenever as a consequence of users' interaction with the system, a new entity is required to be materialized on the screen, the correspondent entity IM²L specification is loaded by the BANCO engine, is localized applying the values given in the LML and TL documents and is materialized by joining it to the Current State Description of the system (the current DOM tree). Therefore, after each interaction, a new Current State Description is reached and materialized and represents the result of the metacommunication between the user and the designer.

The architecture presented in this chapter responds to all the three challenges that emerge from the research context described in Part I of this thesis. It supports the design and development of internationalized interactive software environments localizable to users' culture, role in the context, and digital platform in use. Moreover, it satisfies the requests arising from the co-evolution phenomenon and supports the communication among all the members of the community of interest.

**Part V:
Evaluation of the
architecture**

Chapter 9: Implementing the B-architecture

This chapter describes the concrete implementation of the architecture presented in the previous chapter. In particular, the implementations developed for the case studies presented in Part II are described. My personal contribution to the architecture development started with the second case study presented here, the one in building construction context.

The interaction with the companies and the different domain experts involved in the cases described in this chapter and in Chapter 5, was managed by many researchers in different periods of time. At each step of development, the prototypes were evaluated by using “discount evaluation techniques”, whenever possible by experiments, and by informal meetings with the users. During these meetings, the users were asked to use the current prototype, usability problems were identified by the observation of their activity, and direct feedbacks was collected by unstructured interviews. However, the companies involved did not allow the use of the material collected and it was not possible to use the gathered data to support the evaluation of the architecture in this thesis. However, I used the cases to describe in this chapter how the architecture evolved in time as result of the experiences developed through the different cases.

On the other hand, in the case related to touristic and cultural heritage domain, I had the possibility to perform several evaluations at various step of development and I was

allowed to use the gathered data for the evaluation of my work. Therefore I decided to use that case for the presentation of the evaluation of the architecture in Chapter 10. The other cases described in this thesis are used as examples of implementations in order to demonstrate the feasibility and flexibility of the architecture.

9.1. Factory automation case study

In this case study, several communities of practice were involved, e.g. the software engineers (SE), the domain experts (DE), the company members (Company), the clients of the company (Client), the HCI experts, the company's personnel (Personnel), the assembly-line operators.

Figure 50 illustrates the SSW network of workshops implemented for this case.

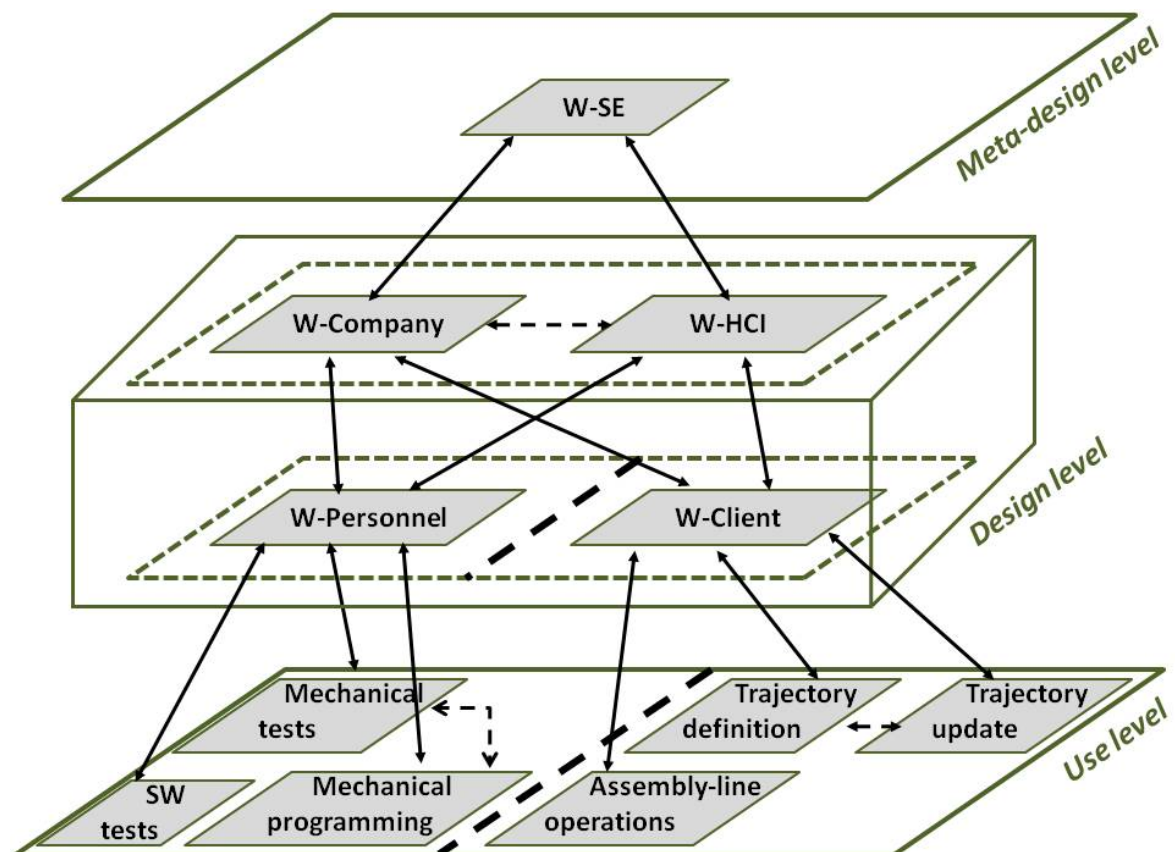


Figure 50. The SSW network for the factory automation case study.

The meta-design level includes the workshop for software engineers, which allows them to generate and maintain all the workshops in the network. The design level includes workshops for designing and adapting the application workshops in accordance with the evolving knowledge and user needs. The use level includes workshops that are used by end users to perform their tasks. The workshops at design level are grouped into two levels.

At the upper sublevel, the W-Company workshop is devoted to the company experts who are in charge of creating all the entities to be managed at the lower plane, while the W-HCI workshop is used by HCI experts to check the human factor aspects of the created entities (e.g. usability, accessibility). At the bottom sublevel, the W-Personnel workshop is used again by the company experts to generate the final workshop, the W-Client, that will be used by the company end users. In our example the W-Client is used by the shop foreman.

Two different types of activity can be identified, the first is the software mechanical design and testing of automation systems and the second is the use of these automation systems in the client factory. The first activity is supported by the use of workshops shaped for company's professionals, e.g. mechanical and software testers and mechanical programmers, while the second activity is performed by the use of workshops devoted to client company's employees, e.g. production managers and assembly-line operators.

In Figure 51 a prototype designed for the assembly-line operator, which is devoted to the control of a pick-and-place robot is shown. In this case, the assembly-line operator needs to have (i) the possibility to choose among different robot's use modalities e.g. automatic, manual, diagnostics (see letter 'a' in the figure), (ii) the power to modify the robot's behavior or the task it has to perform by picking out the tools s/he needs and associate them to it (letter 'b' in the figure), and (iii) the possibility to access tools like annotation, online help, etc (letter 'e' in the figure). The assembly-line operator can observe the behavior of the machine that is shown in the work area (c) and her/his activity is guided by the messages presented in the message area (d).

In what follows, a scenario in which the shop foreman, using W-Client workshop, unwittingly programs the workshop for the assembly-line operator through simple drag-

and-drop activities is described. The assembly-line operator, using her/his workshop can communicate her/his difficulties.

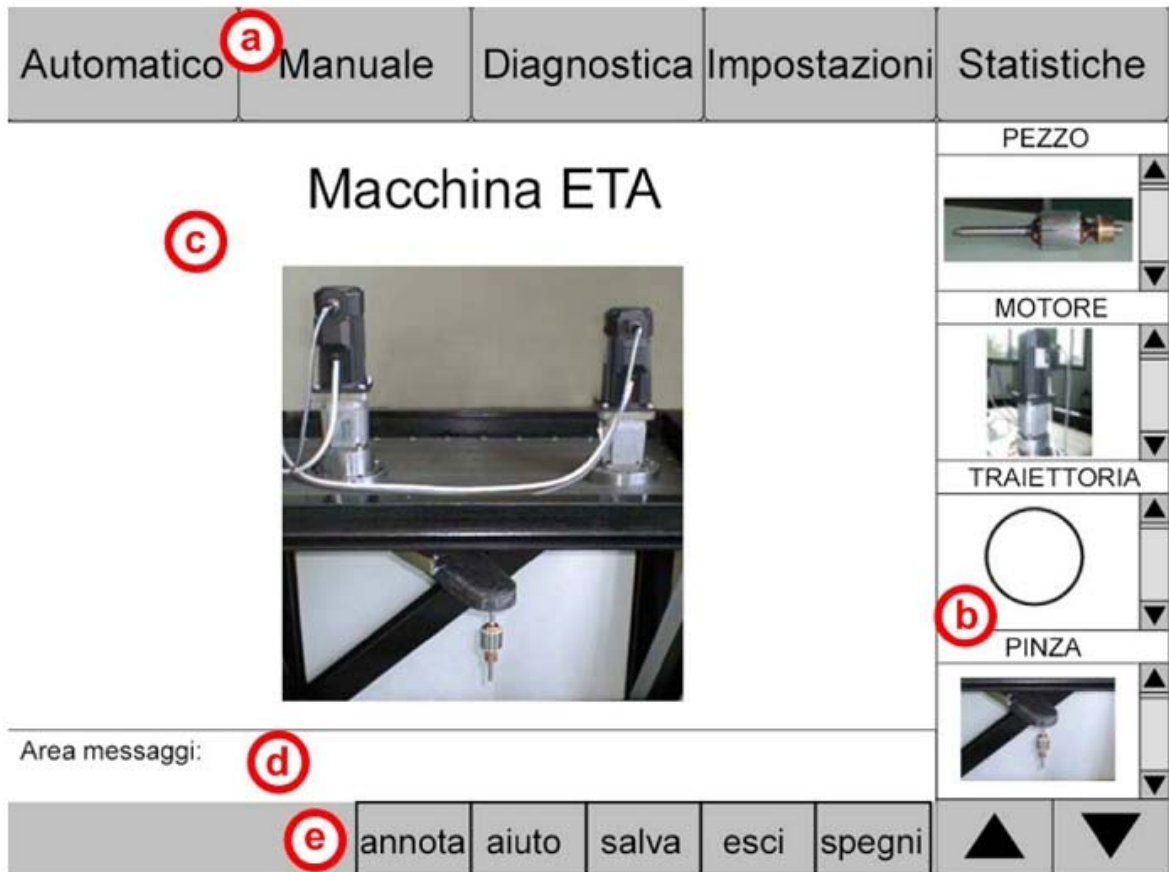


Figure 51. An application developed for a factory automation company. The letters have been added on the screenshot for the sake of explanation in the text (Barricelli et al., 2009c).

Figure 52 illustrates a snapshot of the W-Client at hand, during the interaction of a shop foreman for creating the assembly-line operator workshop shown in Figure 51.

In Figure 52, the shop foreman has already selected the entity “canvas” by choosing it from the menu on the right side: the “canvas” is the background of the workshop that shop foreman is going to create. S/he has already added other entities and now is dragging and dropping the entity “bottoniera di sistema” (system button panel), which will appear on the “canvas” entity in its initial state.

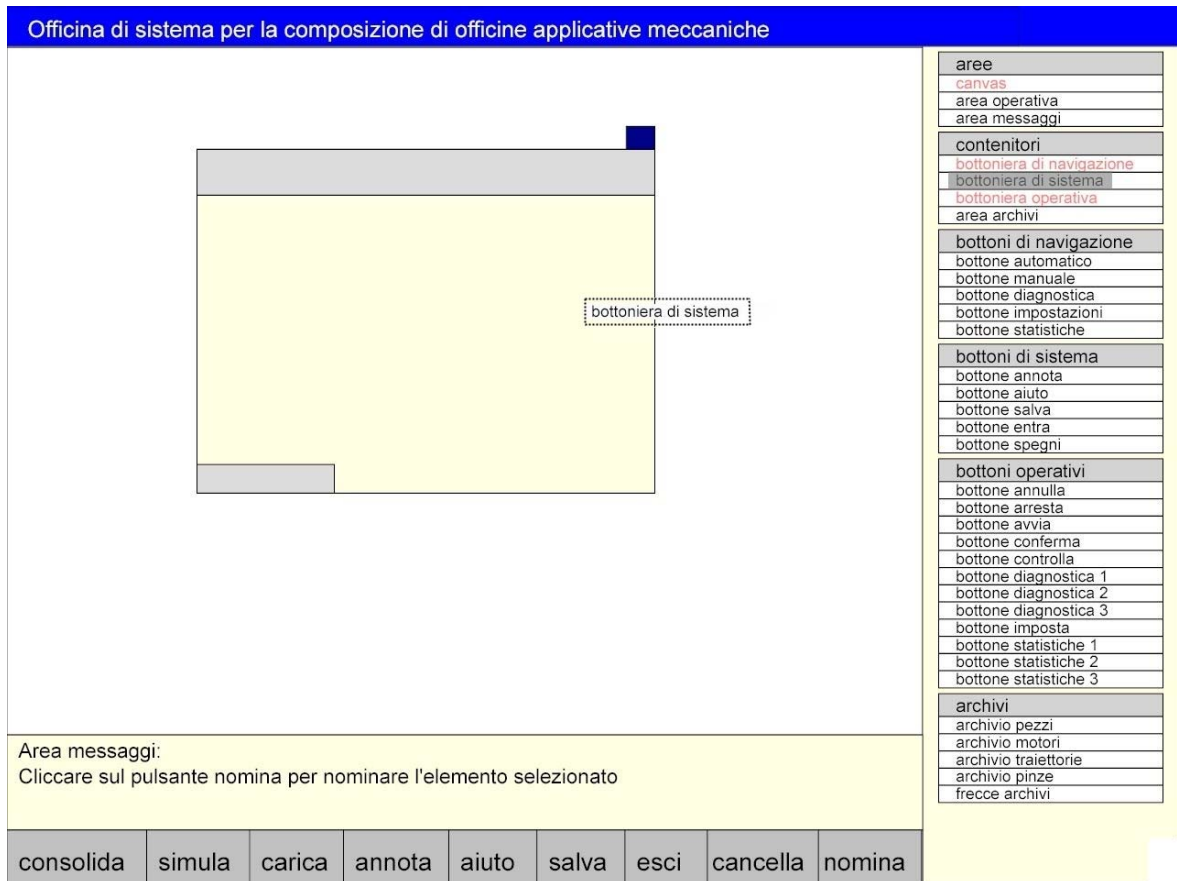


Figure 52. Unwitting programming: the shop foreman drags and drops the entity “bottoniera di sistema” (system button panel) into the canvas representing the background of the assembly-line operator workshop being created.

In Figure 53 the results of the shop foreman’s activity on the assembly-line operator workshop are presented.

S/he is now positioning a new button on the operative button panel. Once the shop foreman considers the realization of the assembly-line operator workshop completed, s/he releases it using the “salva officina” functionality. This functionality creates an instance of the assembly-line operator workshop and makes it available to her/him. The assembly-line operator accesses the new workshop and starts working with it. In performing her/his activity s/he finds usability problems: s/he does not understand how to perform the robot’s diagnostics because the provided strategy is different from the one s/he traditionally adopts in her/his application domain. Therefore, s/he chooses the “annotation mode” selecting the

“annota” button from the “tools” menu. Then s/he selects the “Diagnostica” button. A buttons panel appears, providing tools that permit the insertion of textual annotations.

The system opens the annotation form and s/he annotates her/his observations.

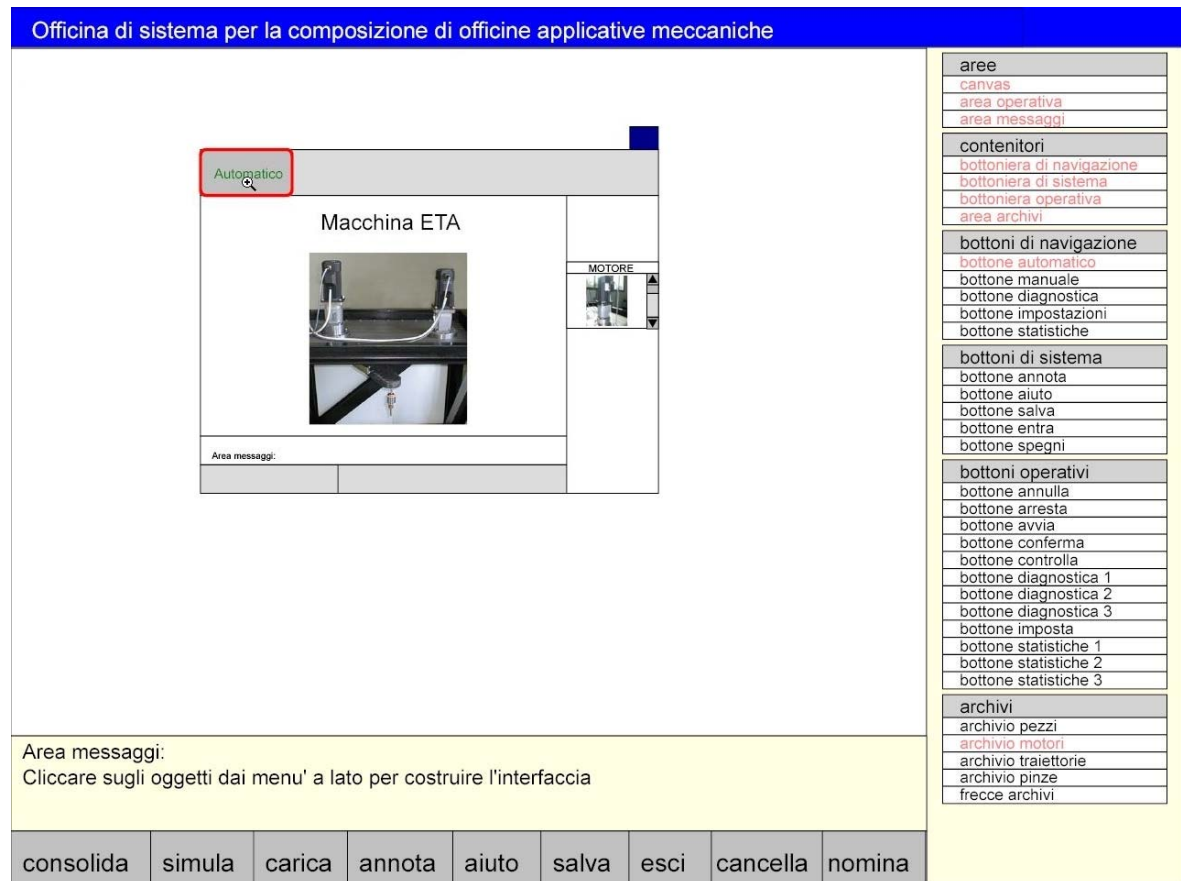


Figure 53. The shop foreman is creating the assembly-line operator workshop shown in Figure 51. The button “automatico” (automatic) is located on the operative button panel.

Figure 54 shows a screenshot taken during the annotation of the assembly-line operator workshop performed by the assembly-line operator.

The shop foreman receives the observations and discusses them with all the other members of the design team to satisfy the end-user requests. Therefore, the design activity is developed as a bargaining activity among the different end users and designers, aimed at balancing requirements on the system that arise from the different points of view.

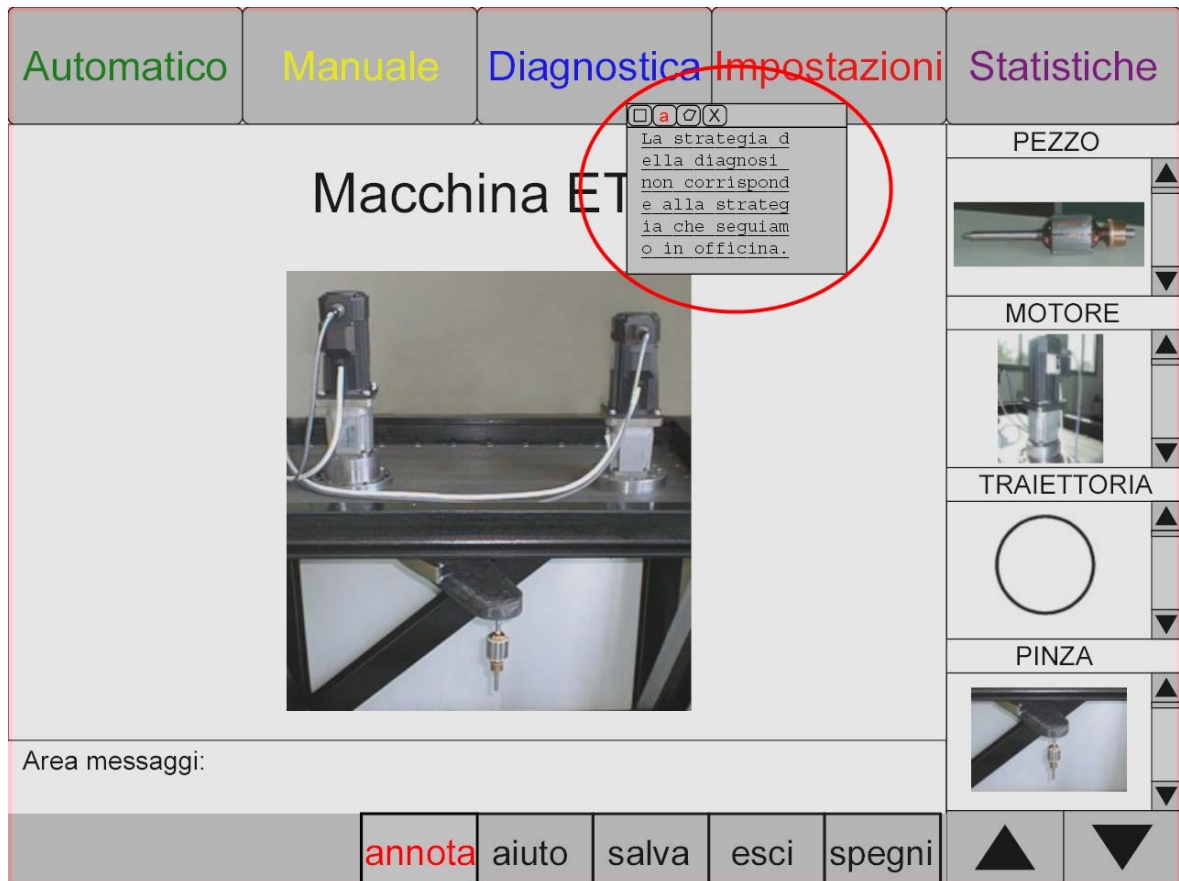


Figure 54. The assembly-line operator annotates the assembly line workshop to communicate her/his problems to the designers. The circle has been added on the screenshot for the sake of explanation in the text.

The factory automation case study highlights two of the challenges described in Chapter 4: system evolution, by involving all the stakeholders in the design and development of the workshops in the SSW network and considering the workshops as perpetual-beta artifact, and communication.

As to communication, it may occur through the different levels from top to bottom and vice versa.

Two types of documents can be exchanged: (i) programs and documents specifying the workshops, for example the shop foreman creates and makes available a workshop to the assembly-line operator; (ii) annotations about usability problems, new user needs or proposed improvements: for example, assembly-line operators can make available to the shop foreman requests for developing new tools to be used in their workshops; whereas at

the use level, assembly-line operators can communicate to HCI experts their problems in understanding the meaning of certain data representations or how to use various tools.

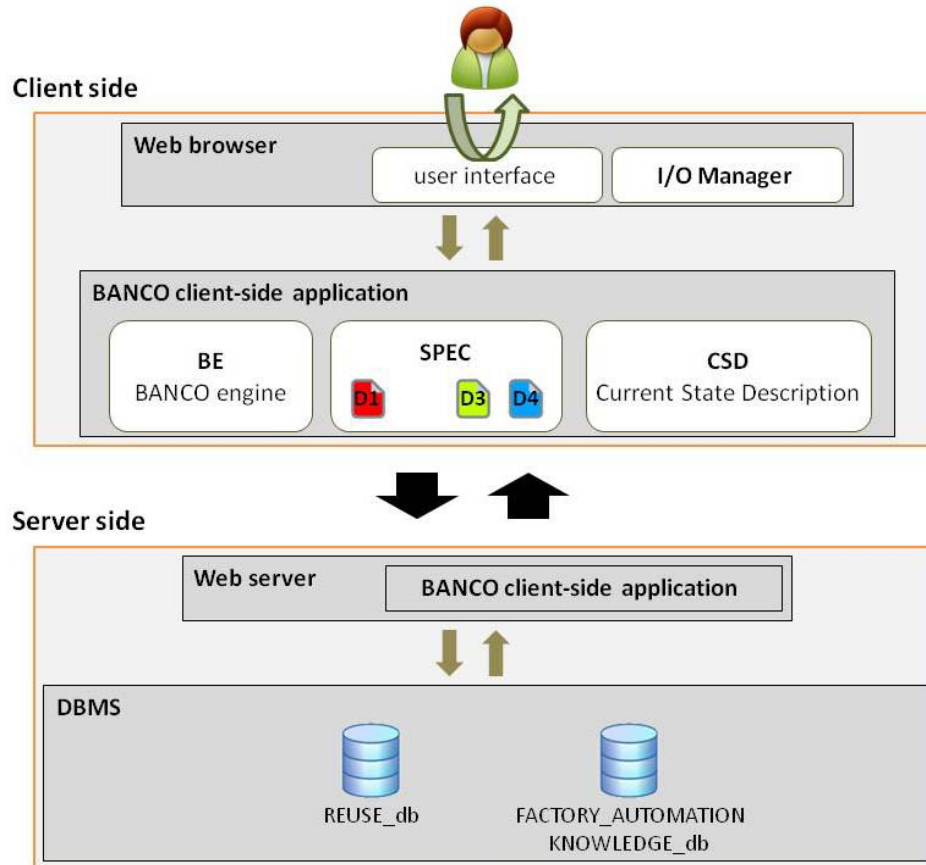


Figure 55. The BANCO architecture implementation for the factory automation case study.

The BANCO architecture implemented for this case study is depicted in Figure 55.

This first case of application of the architecture leads to a definition of its element but still without introducing the localization document. In fact, the environment were implemented considering only the Italian culture.

9.2. Building construction case study

Also in this case study, several communities of practice were involved, e.g. the software engineers (SE), the HCI experts (HCI), the representatives of the end users, and the end users, i.e. technical office operators and foreman on building yard.

Figure 56 illustrates the SSW network of workshops implemented for this case.

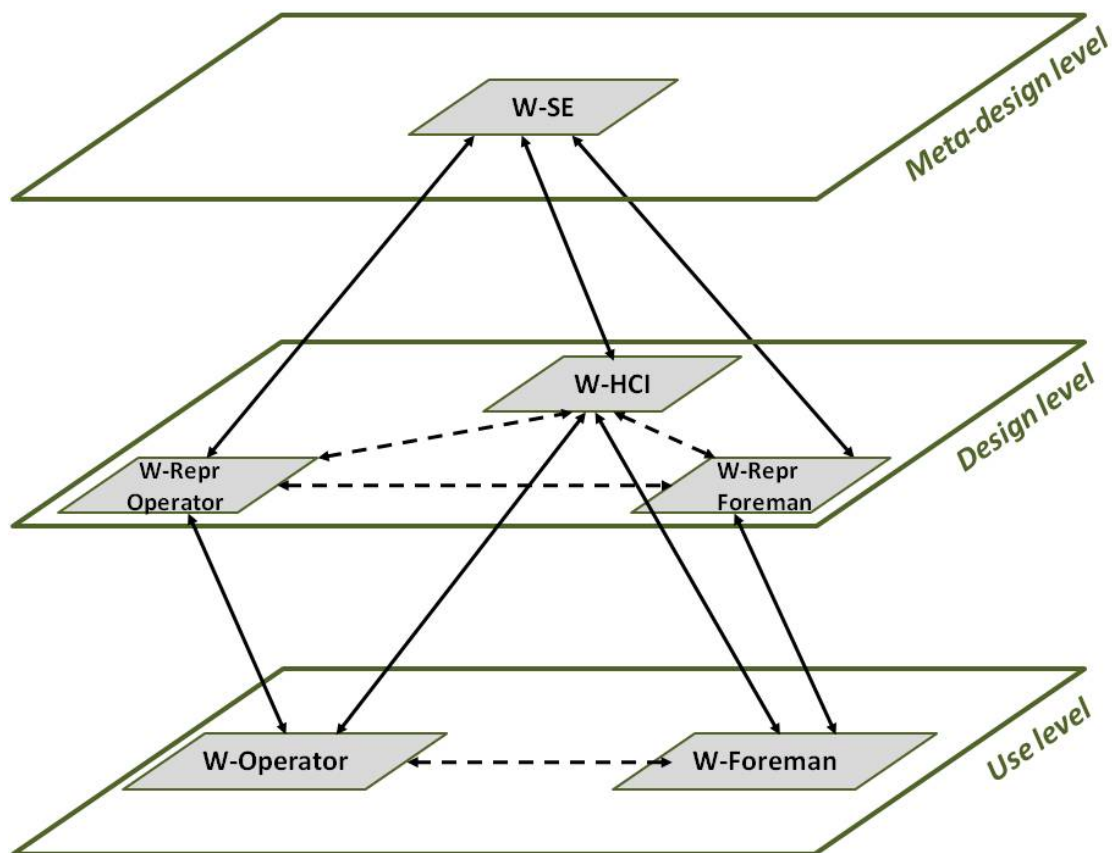


Figure 56. The SSW network for the building construction case study.

The workshops are organized in a three-level network, in which representatives of the users (here called domain experts) and software engineers collaborate to design and develop virtual environments (workshops) customized and tailored for performing domain tasks.

At meta-design level, the software engineers use a system workshop to create further system workshops permitting the HCI experts and the domain experts, to collaborate to the design and development of application workshops. The domain experts are groups of representatives of the technical office operators and of the foremen on the building yard.

At design level, the HCI experts and the domain experts collaborate, using their own system workshop, to design and implement the application workshops.

At the use level, technical office operators and foremen on the building yard use their own workshops to perform their activities.

The scenario (depicted in Figure 23), considers two main locations: the technical office of the enterprise and the building yard. In the technical office an operator interacts with an archive in which data related to the building construction (technical drawings and documents) are stored. The data stored in the archive are accessed through a desktop PC, on which the data are materialized on a large-sized display. On the building yard, the foreman operates with an interactive workshop on a mobile device in which the same data related to the building constructions are represented on a small display. Both the operator in office and on yard, report their activities by annotating the documents on the screen. The foreman loads on the mobile devices a reduced version of electronic documents s/he needs: i.e. the technical drawings of the floor of a building. These drawings sizes are adapted to the device in use, allowing her/him to take annotations on them and to store them via into a temporary archive. An operator on a desktop PC in the technical office updates the original electronic documents and saves the final electronic documents in an archive of the annotated documents.

Figure 57 presents a screenshot of the workshop used by the foreman. S/he can select one of the room maps available in the archive.



Figure 57. The workshop for the foreman presents her/him the documents (room maps) stored in the archive.

On selection, the system presents the map of room 1 (“STANZA 1” in Figure 58) and a transparent layer is loaded over the picture in order to allow the foreman to annotate it using the annotation toolbar displayed on the workshop bottom part.

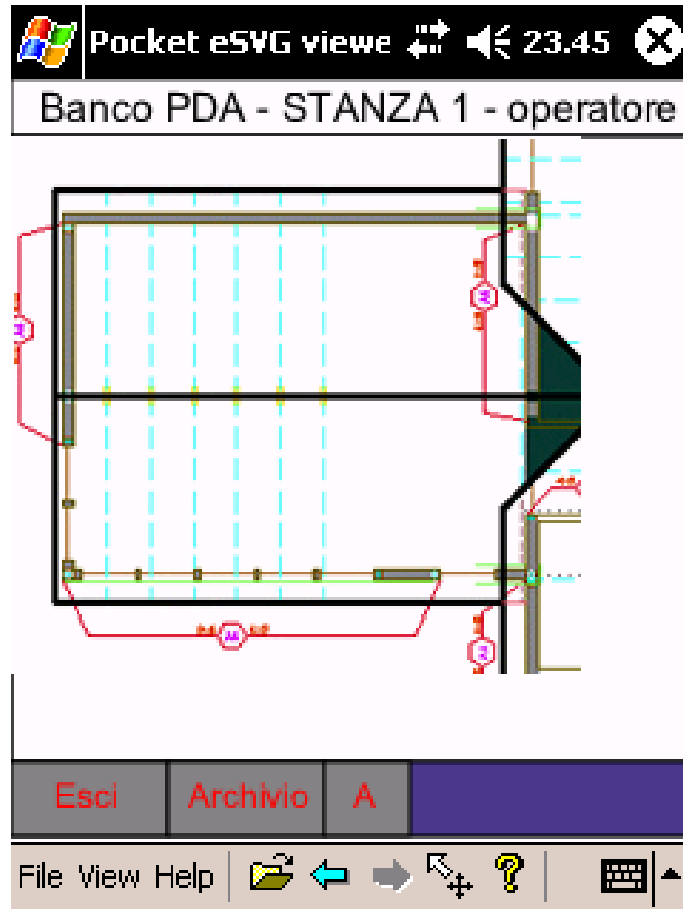


Figure 58. After the selection of a map performed by the foreman, the workshop displays the selected map (“STANZA 1”).

After the creation of the annotation, a visual link (the pencil icon) appears on the map and identifies the annotated area (Figure 59).

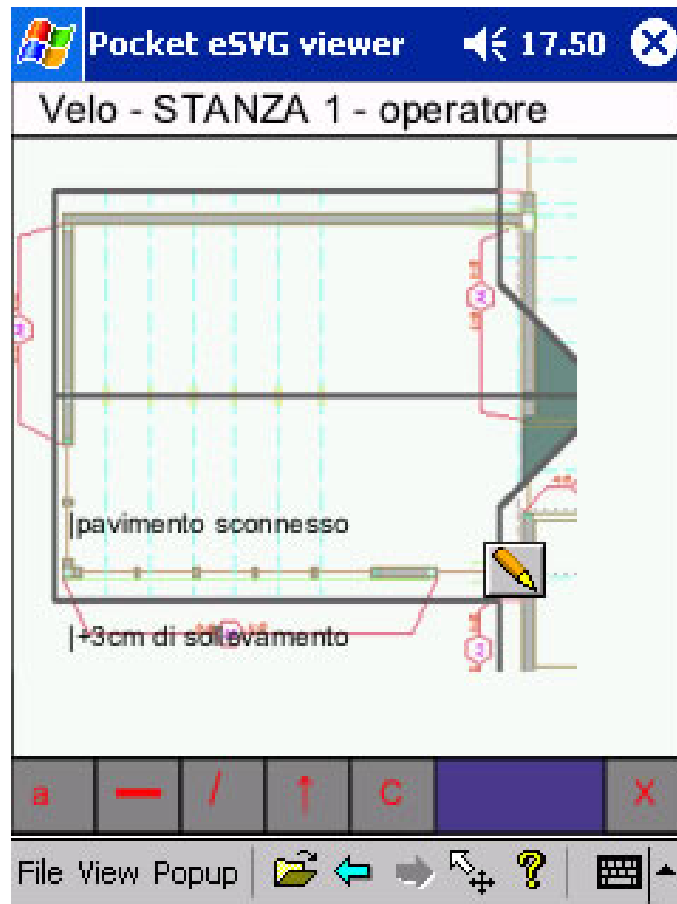


Figure 59. A visual link is materialized on the map to signal the presence of an annotation.

The operator in the technical office accesses the annotated digital maps by opening her/his workshop and by selecting the map annotated by the foreman from the menu on the right (see Figure 60).

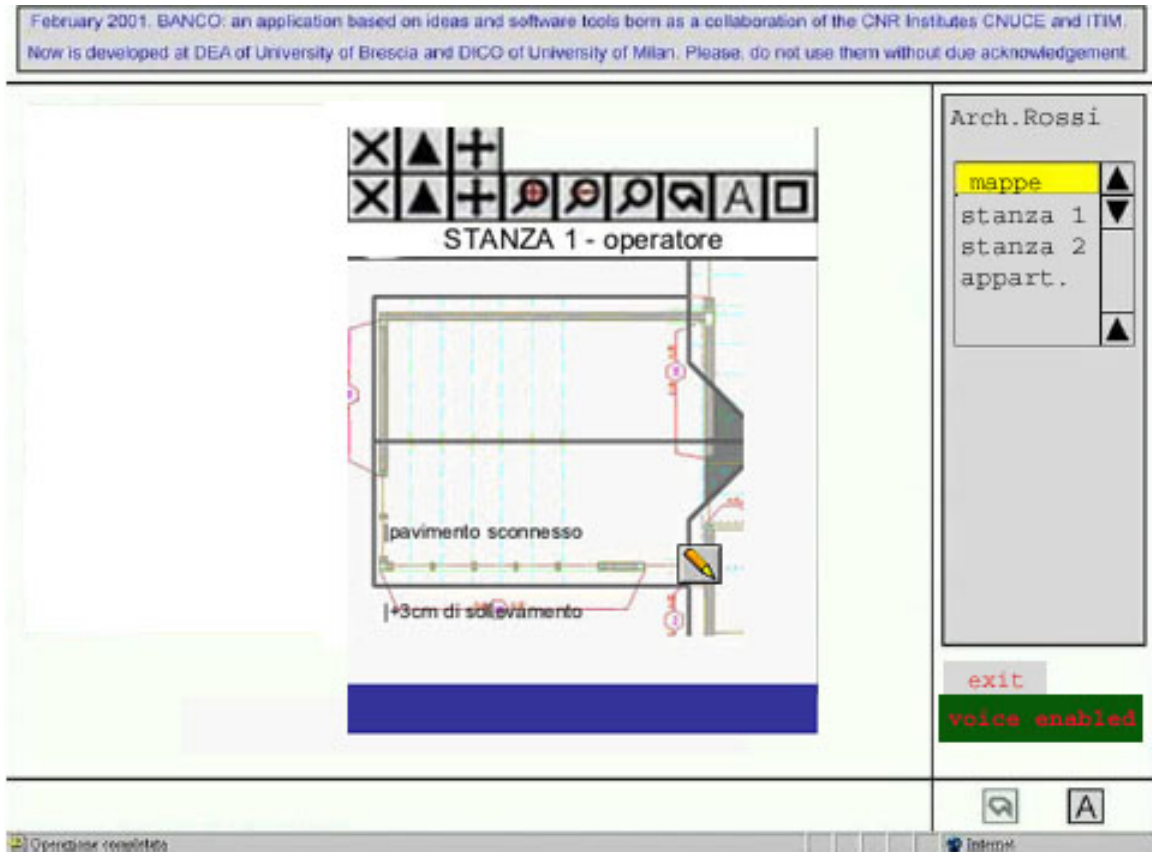


Figure 60. The workshop for the operator in the technical office.

The system materializes on the screen the map requested by the operator. In this workshop, a text-to-speech feature was introduced: using this modality, the workshop notifies about how many annotations are on the map and, when the operator clicks on the correspondent visual links, the annotations content is synthesized. In this way, operator in the technical office can immediately know how many changes have been performed on the map and can select the changes s/he is interested in.

The building construction case study faces at most one of the challenges described in Chapter 4, that is communication. In this case, is in fact very important the communication among the various stakeholders. As described in the scenario, the annotation tool is used by the foreman on the building yard to communicate the changes applied to the building under construction. The system plays the role of mediator between them materializing the data according to their role in the project and to the digital platform in use.

The BANCO architecture implemented for this case study is depicted in Figure 61. This second case of application of the architecture allows the refinement of what defined in the first case study, but still the localization document was missing because the context considered was not multicultural. A new feature was introduced, the text-to-speech modality available for the operator in the technical office. The BANCO architecture was therefore enriched becoming multimodal. The implementation of this feature affected the BANCO engine and the definition of the D3 document of specification (the template document).

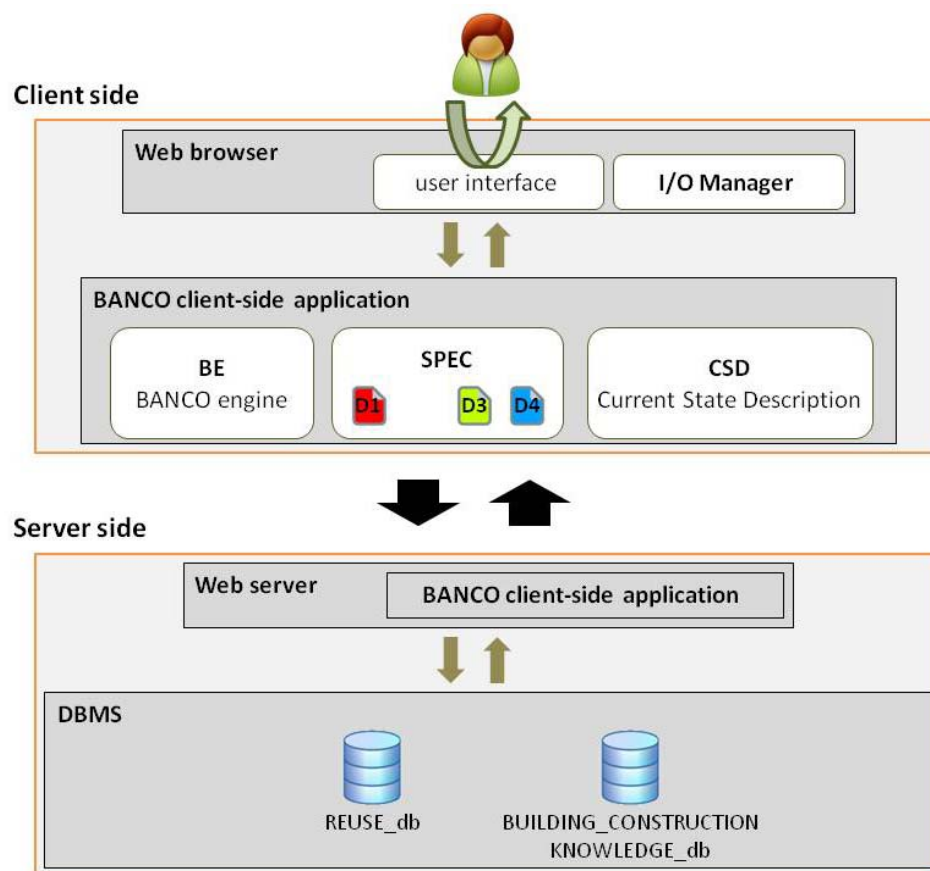


Figure 61. The BANCO architecture implementation for the building construction case study.

9.3. Medical collaboration case study

In this case study, the communities of practice involved were the software engineers (SE), the HCI experts (HCI), neurologists (Neu), and neuroradiologists (NeuRa).

Figure 62 illustrates the SSW network of workshops implemented for this case.

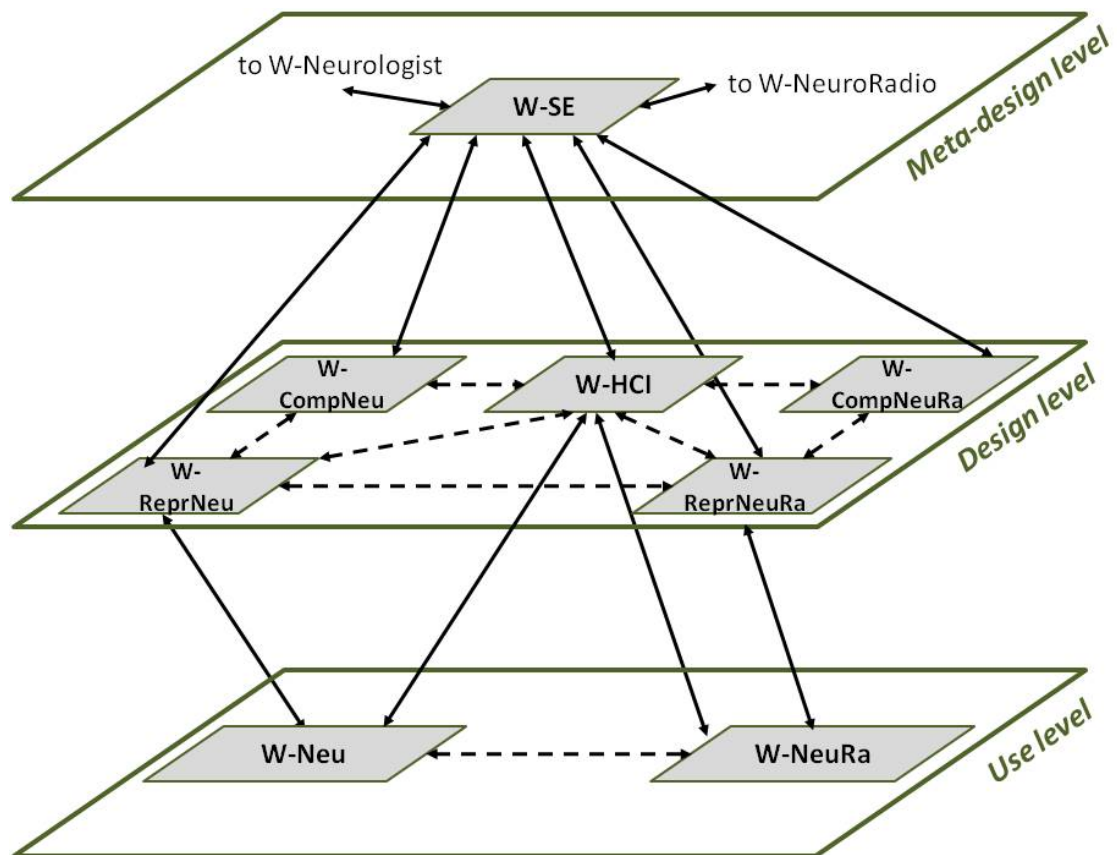


Figure 62. The SSW network for the medical collaboration case study.

At meta-design level, software engineers using a system workshop, called W-SE, create customized workshops to be used by the experts at design level (HCI experts, Neurologists, and Neuroradiologists).

At design level, representatives of neurologists and neuroradiologists using two different system workshops (W-ReprNeu and W-ReprNeuRa) create and maintain the application workshops devoted to the use level. With other two system workshops (W-CompNeu and W-CompNeuRa) they create and maintain components to be used in the other system workshops for design activities. At design level also the HCI experts collaborate using their workshop (W-HCI). All these experts collaborate in order to bring their own knowledge and expertise in the creation and evolution of the two application workshops used at use level.

At use level, the workshops created at design level are used by the neurologists and neuroradiologists in order to perform their activities and to exchange between them information and opinions using the annotation tool. The two workshops differ in that they offer different tools, according to the specialty of the physician.

Figure 63 shows a screenshot of the system workshop W-ReprNeuRa: the neuroradiologist has partially composed the application workshop devoted to the neuroradiologists by selecting, from the repositories on the right side of the screen, the canvas on which the application workshop is composed, and other components that could be useful for the end user, i.e. the other neuroradiologists who operate at use level. The neuroradiologist is developing a workshop by visually interacting with the system workshop and is not writing any textual code.

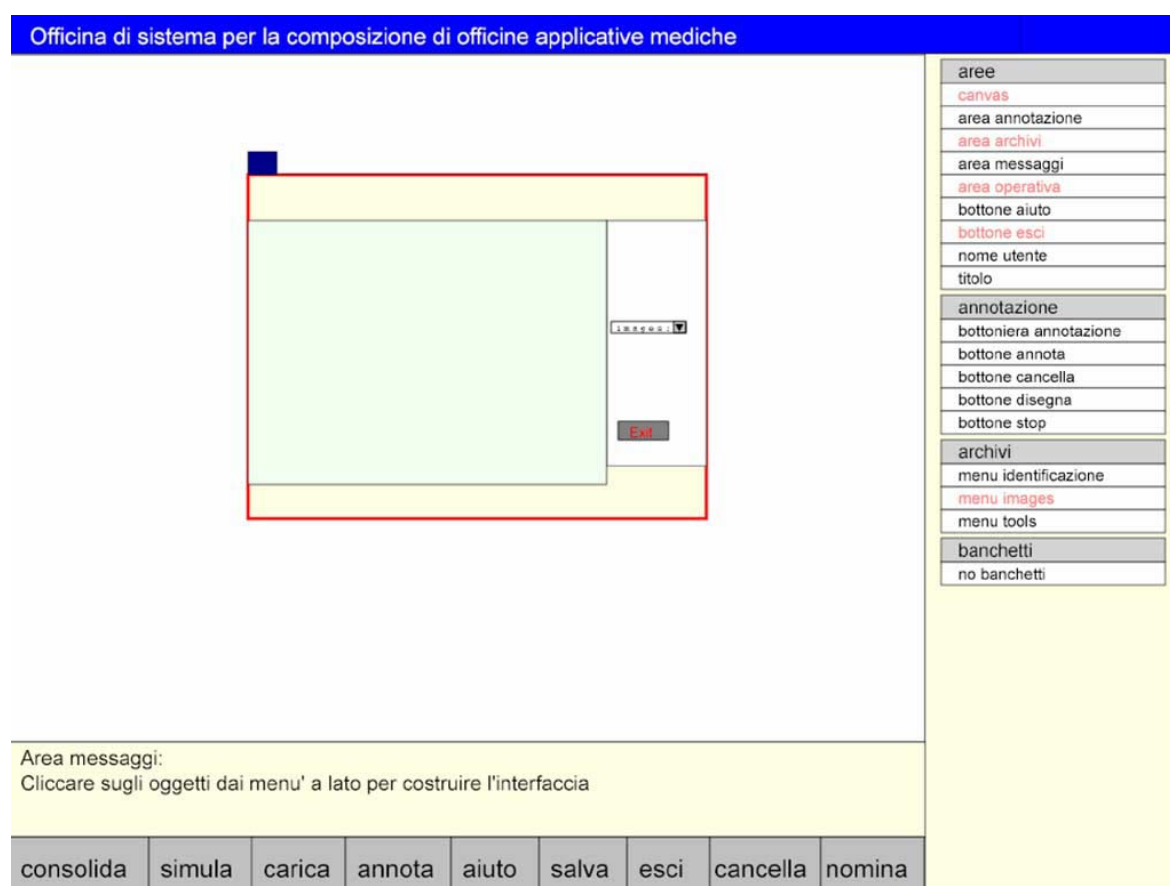


Figure 63. System workshop W-ReprNeuRa used by a neuroradiologist who creates the application workshop devoted to the other neuroradiologists at use level.

In Figure 64 the system workshop called W-CompNeuRa used by a neuroradiologist is depicted. The neuroradiologist creates the workbench to be used by neuroradiologists who operate at use level. As for the W-ReprNeuRa workshop (Figure 63), the neuroradiologist can select objects from the repositories and drag and drop them in the work area to create, in this case, a workbench (i.e. a component of the final workshop). After a workbench is saved, it appears as one of the benches available in the corresponding repository in the system workshop W-ReprNeuRa presented in Figure 63.

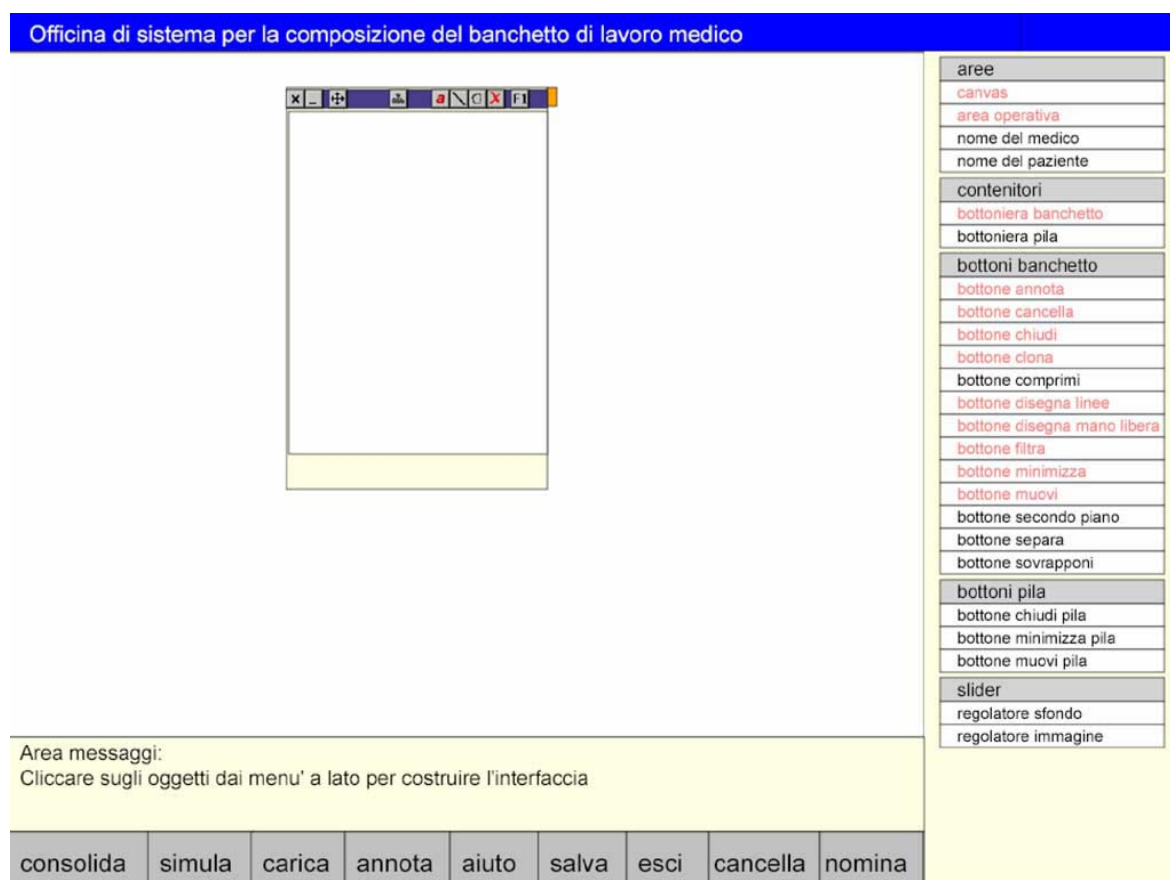


Figure 64. System workshop W-CompNeuRa used by a neuroradiologist who is in charge of creating components for the workshop devoted to the neuroradiologists who operate at use level.

The main contribute of this case study to the BANCO architecture was the introduction of the localization document. In fact, the workshops created for the end users operating at use level are completely localized for their culture and not only for their role and digital platform.

As an example, two instances of workshops used at use level by two neurologists are shown in Figure 65 and Figure 66. Figure 65 shows the workshop for an Italian neurologists, the organization of the content in the interface follows the culture and role of the user. The workshop is also localized to the digital platform in use that is a desktop PC.

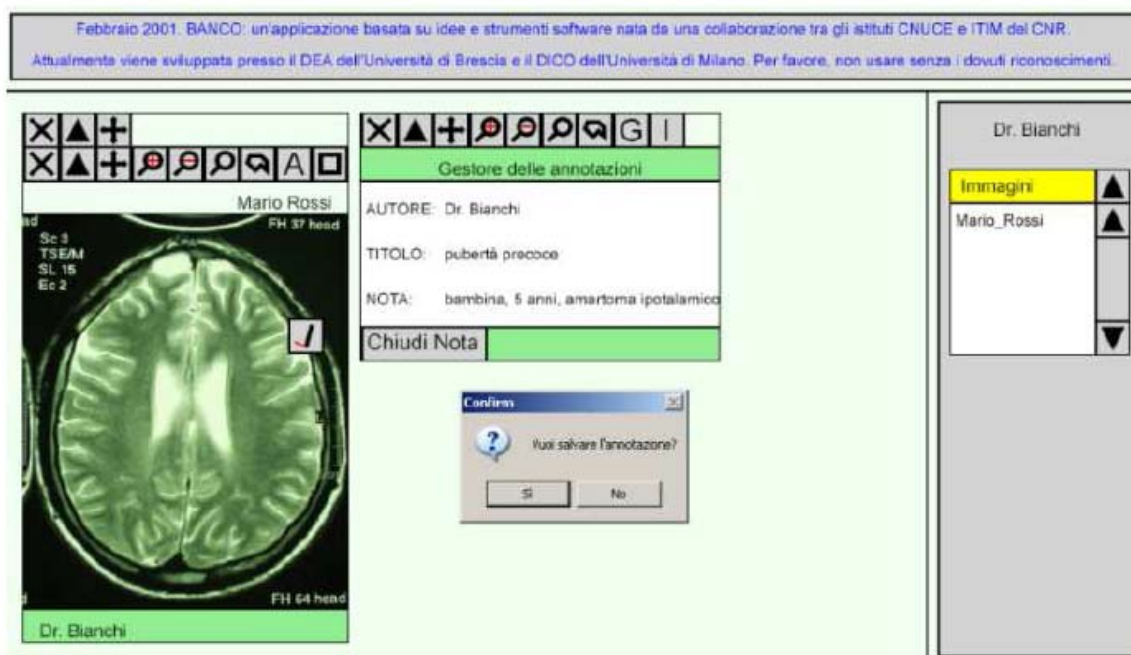


Figure 65. The workshop for an Italian neurologist. The interface is localized to her/his culture, role and digital platform in use.

In Figure 66, the workshop depicted is for a Hebrew neurologist and the interface organization follows her/his culture and role rules. Also this workshop is localized for the use on a desktop PC. In this case, the visual entities on the screen are materialized according to the reading/writing direction of Hebrew mother tongue people, that is from right to left and from top to bottom. As explained in Chapter 3, the localization of a software system does not affect only the translation of the texts, but also the organization of the elements and the adaptation of the contents.



Figure 66. The workshop for a Hebrew neurologist. The interface is localized to her/his culture, role and digital platform in use.

From what presented in this chapter, it is evident how the medical collaboration case study faces all the three challenges described in Chapter 4: system customization, system evolution and communication.

Figure 67 describes the implementation of the BANCO architecture for the medical collaboration case study. This case led to the definition of the localization document, considering not only the role of the end users but also their culture.

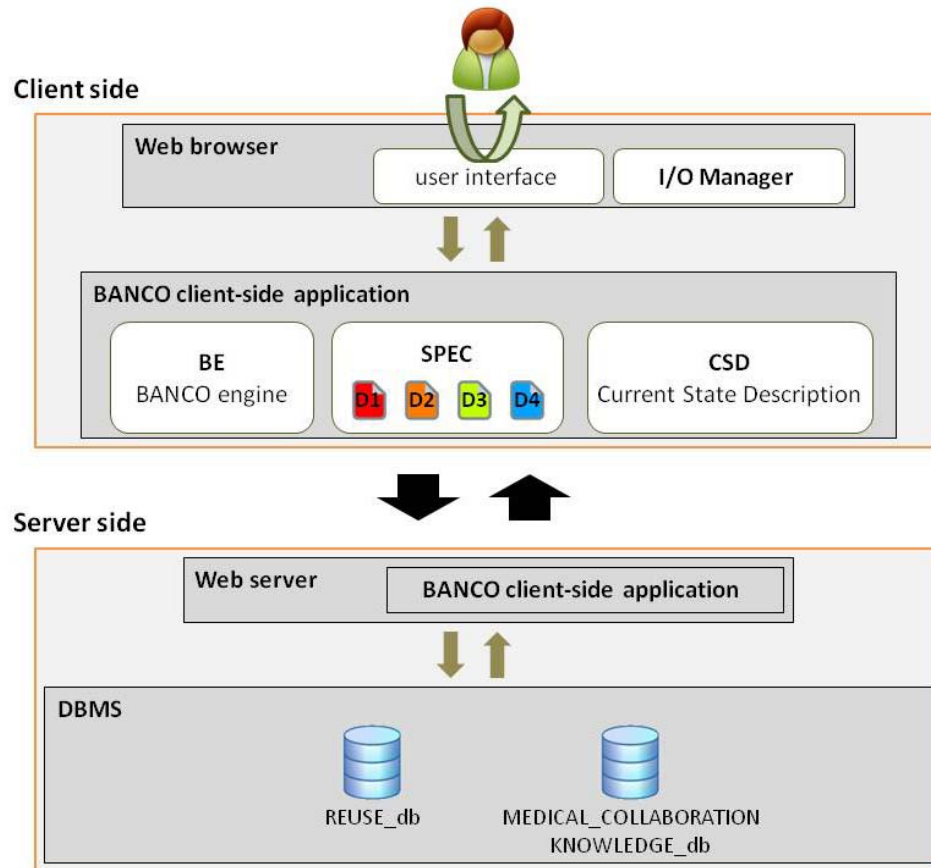


Figure 67. The BANCO architecture implemented for the medical collaboration case study.

9.4. Tourism and cultural heritage case study

This case study involves the following communities of practice: the software engineers (SE), the domain experts (DE), the company members (Company), the HCI experts, stakeholders who manage the content publishing (Publisher), and tourists.

Figure 68 illustrates the SSW network of workshops implemented for this case.

At meta-design level the software engineer uses her/his workshop (W-SE) to design the workshops for the other levels in the network.

At design level HCI experts, publishers, and domain experts collaboratively design the workshops to be used at the use level by the tourists. Each CoP is provided with specific workshops that offer the tools adapted to the role of their members.

At use level the tourists are provided with workshops that are localized to their culture, role and digital platform in use.

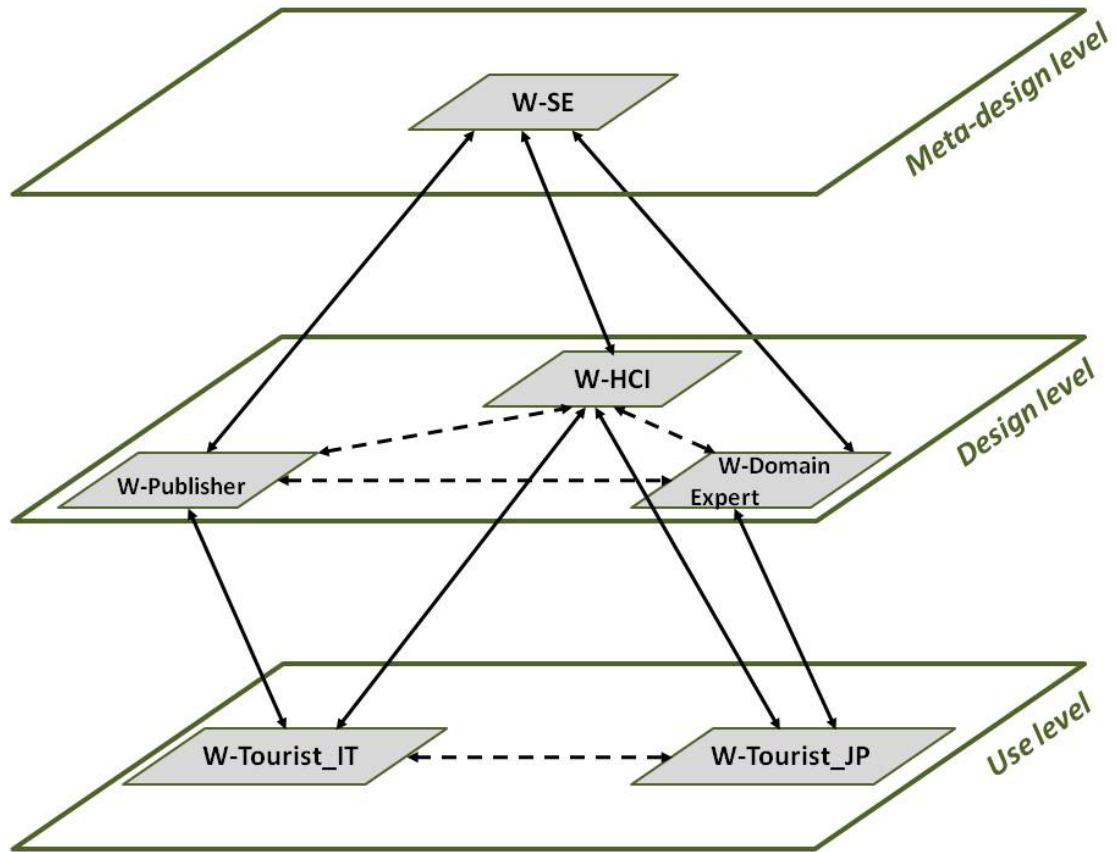


Figure 68. The SSW network for the tourism case study.

The design level represents the core of the system network because there several workshops are used by the domain experts to both create a shared knowledge base related to Valchiavenna region and develop and maintain the workshops to be used by the tourists to access the knowledge base and to leave annotations.

The shared knowledge base is the result of the social interaction among tourists belonging to different cultures and domain experts. They all jointly enrich the knowledge base with certified or personal annotations, which illustrate relevant topics referring to the maps or with impressions from their travels in the region represented by the map.

Two examples of workshop used at design level are given in Figure 69 and Figure 70.

In Figure 69, the workshop used by the publisher is depicted. The publisher is in charge of associating the contents created by the domain experts to the maps in order to make them available to the tourists.

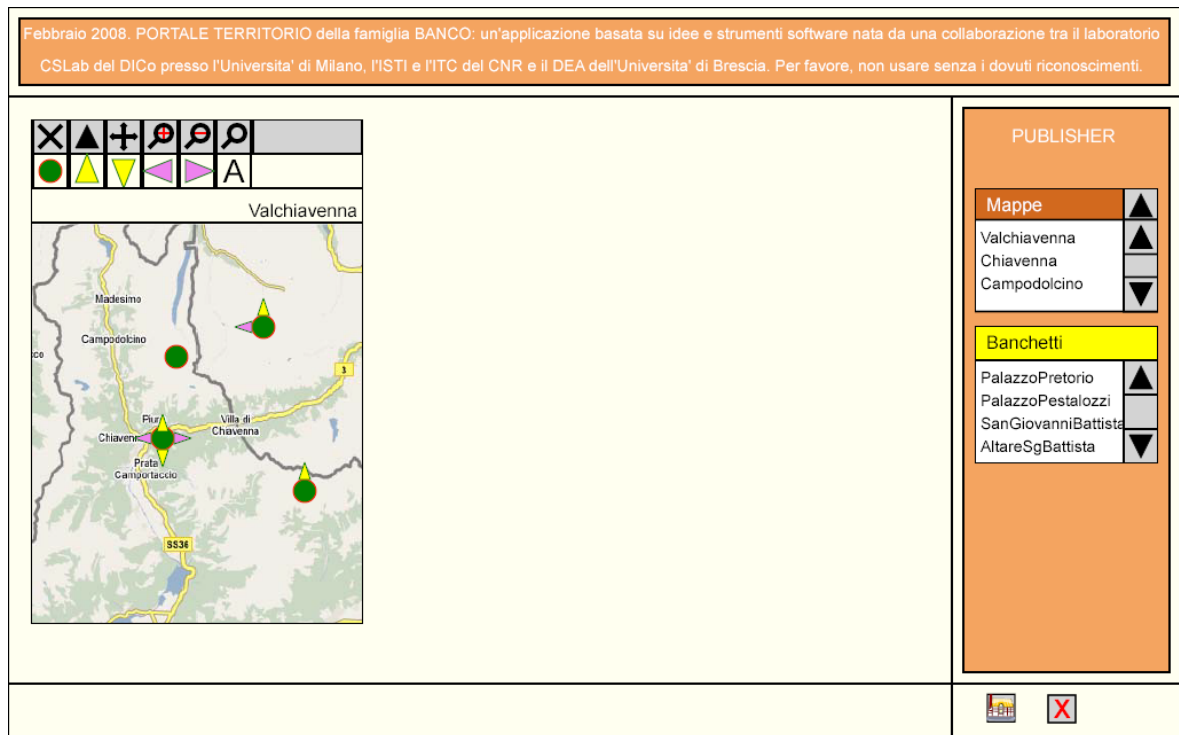


Figure 69. The workshop used by the publisher in order to associate to the maps the contents created by the domain experts. The publisher is also in charge of moderate the annotations created by the tourists.

The domain experts involvement in the collaborative project is not just related to their knowledge sharing, but also to the development of the workshops for the tourists. The domain experts are in fact required to develop some of the components of the tourists workshop by choosing the functionalities that may be used by them in virtually exploring the Valchiavenna region. Figure 70 shows the domain experts' workshop where s/he can select the functionalities to be added by a set of repositories available on the right side of the screen. After the composition of the workshop, it is saved and used at use level and its visual materialization will be localized using the different localization documents available in the architecture.

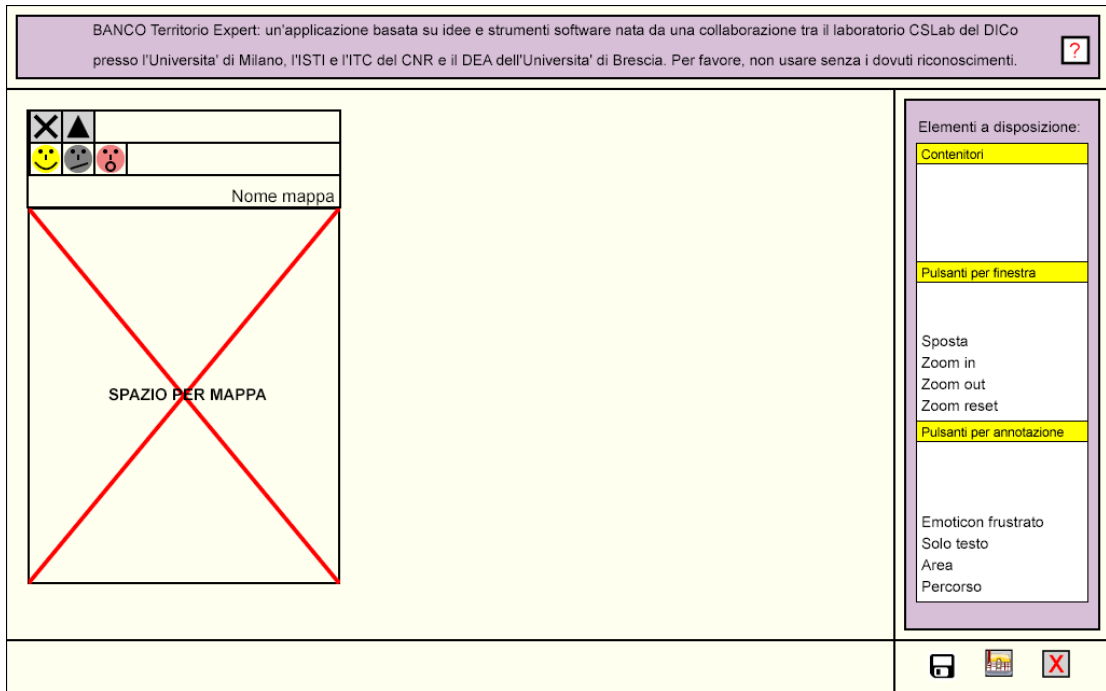


Figure 70. The workshop used by the domain expert in order to develop components for the workshop for the tourists. In particular, in this picture the domain expert is building a workbench by selecting operators from the available repositories.



Figure 71. An Italian tourist accesses the Valchiavenna map using her workshop and creates an annotation using an emoticon that expresses appreciation respect to the annotated point of interested.

The Sistema Culturale Valchiavenna case study faces all the three challenges described in Chapter 4: system customization, system evolution and communication.

As to system customization, the localization performed by the system has been refined and considers also the meaning that colors convey in the different cultures. This implementation of the architecture responds to all the problems highlighted in Chapter 3 and implements what proposed by Yeo (1996).

Two instances of tourist workshop are shown in Figure 71 and Figure 72.

The first instance (Figure 71) is the one devoted to an Italian tourist. She accesses the Valchiavenna map and creates an annotation related to a point of interest that she considers nice to be visited. To express her mood about the point of interest she can choose among four different emoticons representing four moods: appreciation, surprise, disappointment and sense of danger.



Figure 72. A Japanese male tourist accesses the Valchiavenna map using his workshop and creates an annotation using an emoticon that expresses appreciation respect to the annotated point of interested.

The same action is performed by a male Japanese tourist (see Figure 72). He accesses the Valchiavenna map and after choosing an emoticon that better represents his mood, he creates an annotation on a point of interest.

The two tourists are able to use the same scale of liking to assign a mood to their annotations. However, the materialization of the emoticons differs according to cultural rules. In Figure 73 the four emotions appreciation, surprise, disappointment, and sense of danger are represented with different shapes and associated to different colors. The specific case of Japanese culture is representative because the shape of an emoticon should change according to the gender of the user. In fact, the traditional Japanese culture, prohibits to women to show their teeth while smiling. Therefore, for the SCV case, two different localization documents have been developed, one for female Japanese tourists and one for male Japanese tourists.













SCALE OF LIKING					
Emotion	Italian colors	Italian Shape	Japanese colors	Japanese male shape	Japanese female shape
Appreciation	yellow		Green		
Surprise	lightcoral		Lightcoral		
Disappointment	Gray		Blue		
Sense of danger	Red		Red		

Figure 73. The scale of liking used in the SCV case study. Four different emotions are materialized with different shapes and colors according to Italian and Japanese cultures.

As to system evolution, the participation of all the stakeholders to the development of the various workshops used in the network allows to apply a collaborative and at the same time end-user development approach.

As to communication, the annotation tool is used by a) the domain experts to contribute to the creation of a certificated shared knowledge base, b) by the tourists to exchange opinion about the Valchiavenna region and c) by all the stakeholders in the network as tool for the

communication among the different levels in the network, in order to report usability problems of errors in the system.

Figure 74 illustrates the implementation of the BANCO architecture for the tourism and cultural heritage case study. This case led to the refinement of the specification documents and to the introduction of the use of the eXist XML native database. The indexing functionality has been introduced and also the glossary has been adopted (as explained in Chapter 8).

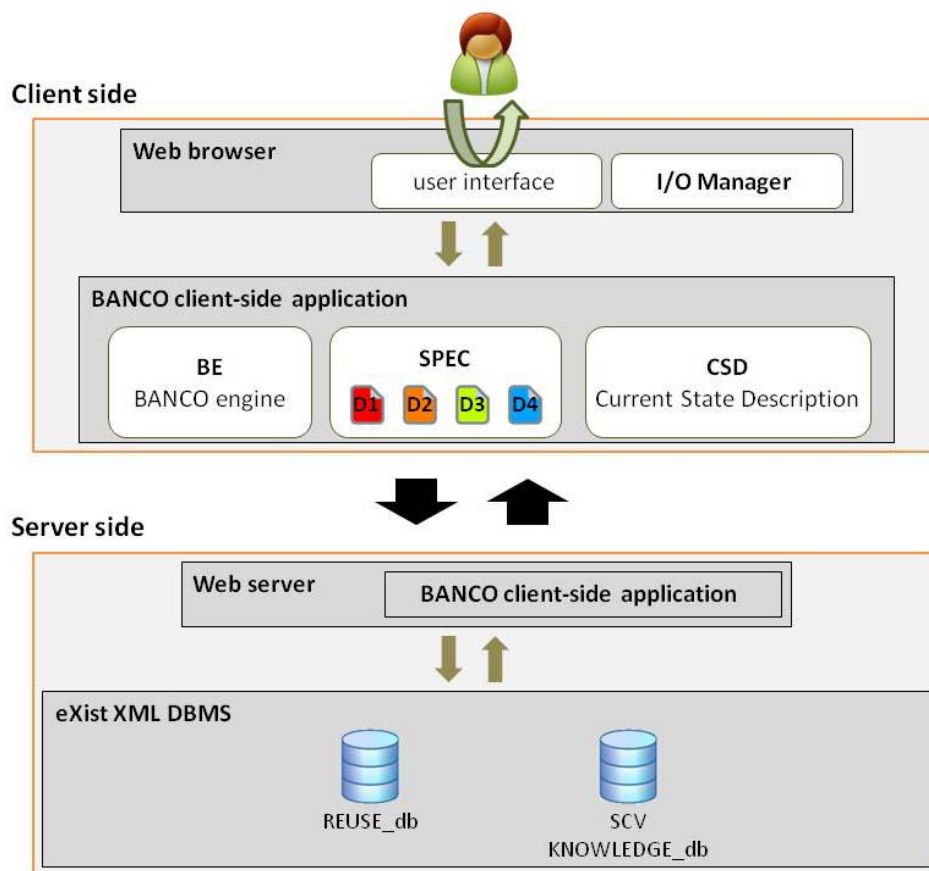


Figure 74. The BANCO architecture implementation for the tourism case study.

This case study is the most representative (and recent) for the BANCO architecture definition and implementation and therefore has been used for the evaluation of the architecture presented in Chapter 10.

9.5. Workflow management case study

The communities of practice involved in this case study are the software engineers (SE), the domain experts (DE), the HCI experts, the workflow designers, and the workflow operators.

Figure 75 presents the SSW network of workshops implemented for this case.

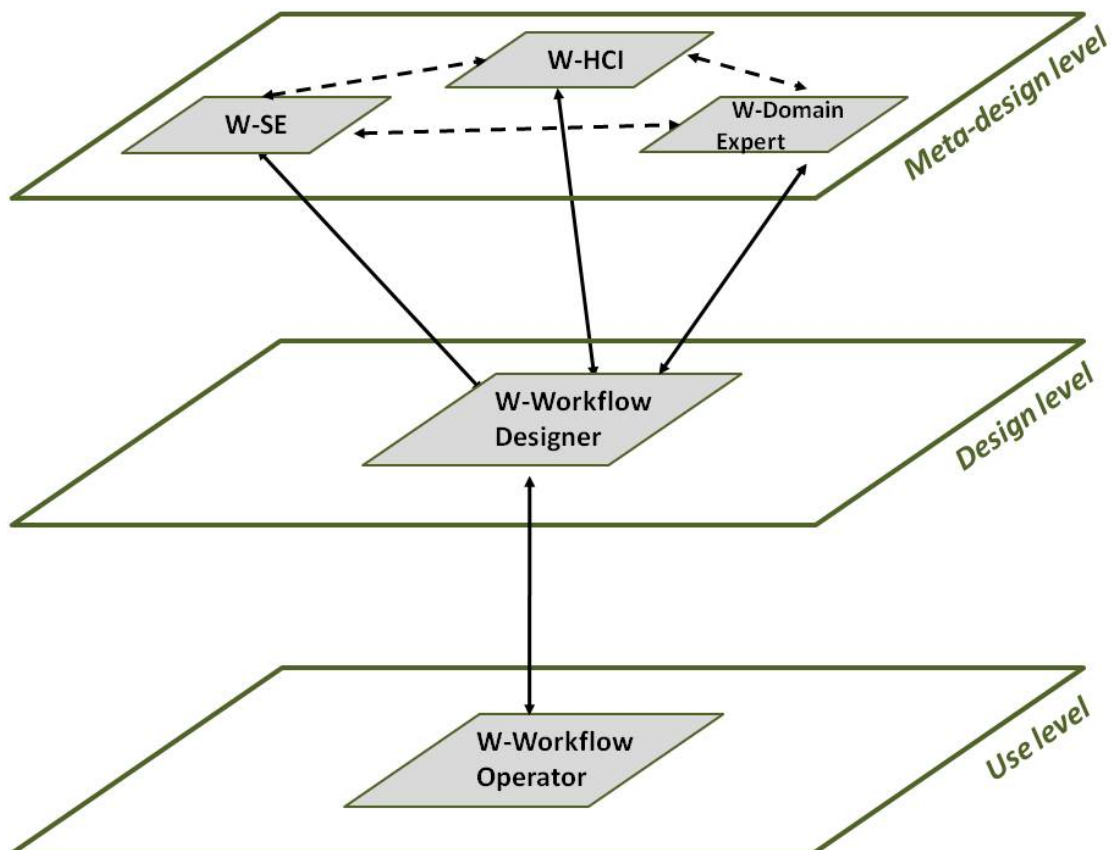


Figure 75. The SSW network for the workflow management case study.

At meta-design level, the software engineers and the HCI experts are in charge of developing the workshop to be used by the workflow designer at design level. On the other hand, the domain experts define the overall structure of the workflow providing to the workflow designer a set of documents to be used as support of her/his work. They use tools for task analysis to represent in a structured way concepts and activities that constitute the workflow, without referring to implementation issues.

At design level, the Workflow Designer uses the TMS Editor workshop by which s/he transforms the task analysis document, created by the domain experts, in a description of the needed components (tComponents) and the relationships among them.

At use level, the Workshop Operators use the TMS instances developed at the above levels in order to perform their work activities.

The goal of the network of workshop in this case study is a) the design and check by visual composition of a workflow, b) its visually validation and execution, and c) its execution and use.

This case study is currently ongoing and just a first workshop, the one for the workflow designer, the TMS Editor, is under development. TMS Editor follows a semi-automatic approach. Its goal is not to automate the skills of the end users, but to assist them in their activity, permitting them to use the knowledge and expertise they possess.

The aim of the TMS Editor is to support the design activity of the Workflow Designers allowing them to exploit their tacit knowledge and expertise, without boring them with technical details about the language used to implement, store and transmit the final workflow document. The TMS Editor interface is designed so that the Workflow Designers can map their mental models of the activities to be performed into visual commands and widgets (Valtolina, 2008). The visual widgets are used to show tComponents and their relations in a graphic way translating the semantic description of the tComponents and their relations into visual forms. On the other side, this semantic description enables a mapping between the WSDL (Web Services Description Language) (WSDL) interface of the tComponent with a Conceptual Reference Model (CRM) describing its behavior, goal and functionality. The CRM of a tComponent is a set of classes and properties, expressed in OWL (Web Ontology Language) (OWL), indicating entities and relationships for describing the tComponent's usage field, its inputs and outputs, the interaction style used and a description of its specific algorithmic behavior. Moreover each CRM is extended using specific concepts related to the information domain in which the workflow takes place. These concepts in CRM are used to map tComponents usage field and behavior in specific domain concepts in order to explain, adopting a

semantic way, the function and the role of each tComponent. The mapping between each WSDL description and the related CRM of each tComponent is carried out with a RDF file while the mapping between the CRM description and the visual widget representing it, is achieved in the TMS Editor by means of its visual interface. Therefore, exploiting these visual representations of each tComponent, the Workflow Designer is able to design a workflow through the integration of heterogeneous tComponents in a particular order and under certain conditions. To support this activity the TMS Editor has to adopt specific techniques of composition and integration management services. To this end, an Orchestration Engine (OE) translates the composition defined by the Workflow Designer using the TMS Editor interface, into a BPEL4WS (BPEL4WS) workflow document describing the correct sequence of operations in the workflow following the structure defined at meta-design by domain experts. The Semantic Search Engine (SSE) retrieves the tComponents useful to compose the final structure of the workflow following the directives of the Workflow Designer interacting with the TMS Editor (see Figure 76).

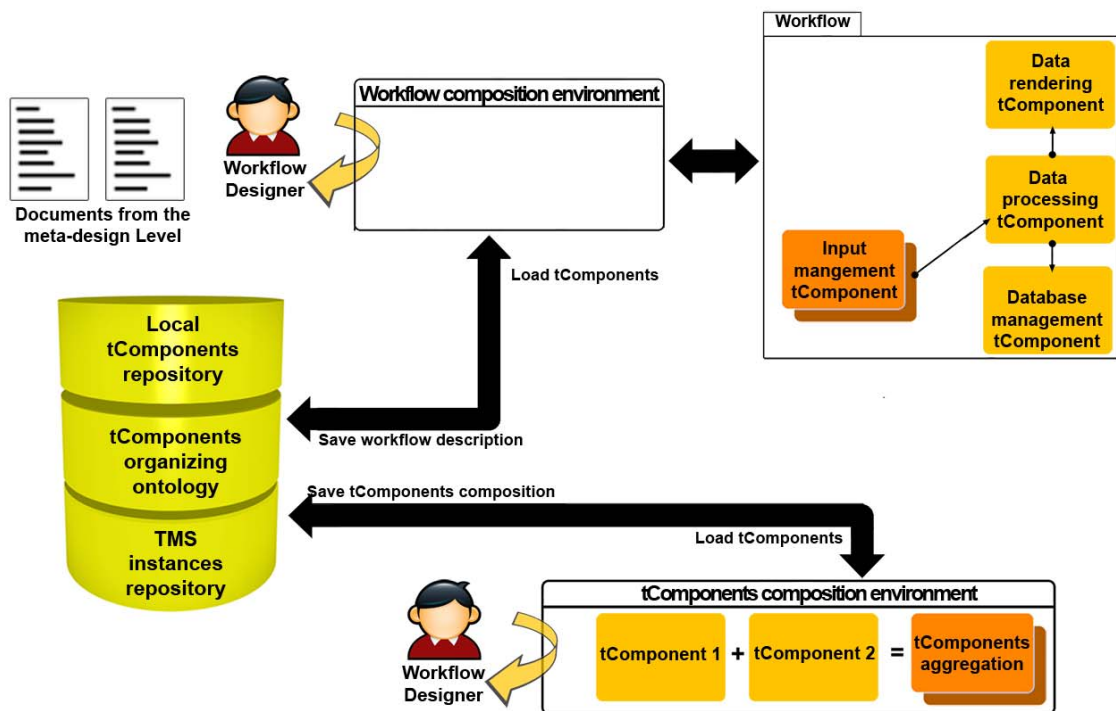


Figure 76. The Workflow Designer-TMS Editor system for semi-automatic workflow composition

These tComponents are gathered by a knowledge base composed by a set of archives made available by various tComponent providers. The tComponents provider, willing to share its tComponents, must first perform a UDDI (Universal Description, Discovery and Integration) registration of the tComponents on a tComponents Provider Registry.

The registration consists of three different elements: 1) a WSDL description of the tComponent to be invoked, describing the syntactical and computational-based aspects about the tComponent; 2) a Conceptual Reference Model (CRM) of the tComponent; 3) a RDF description that maps the CRM onto the WSDL description of the tComponent. Interacting with the SSE, Workflow Designers can search the more suitable tComponents, according to the context of use of workflow defined at meta-design. Then they trigger the orchestration activity for producing the final description of the workflow process as a BPEL4WS document. This document is represented on the screen as a sketch of the widgets representing the tComponents and their relations.

The workflow management case study is still ongoing therefore the changes in the architecture needed for this case are not yet completely defined. However, up to now the most important contribution to the architecture (depicted in Figure 77) is the introduction of the use of external Web services that are used by the workflow designer to compose the workflow definition.

Moreover, the BANCO architecture finds in this case study a very good example of application because supporting heavily component-based developing approach represents a good adoptable choice. Component-based development is in fact an approach that highly supports end-user development (Ginige et al., 2005; Stevens et al., 2006; De Silva et al., 2009).

The study on ERP performed by Dittrich et al. (2009), led them to some basic assumptions about software engineering. In particular, the attention is on the validity of component-based programming to support the customizability of the resultant system.

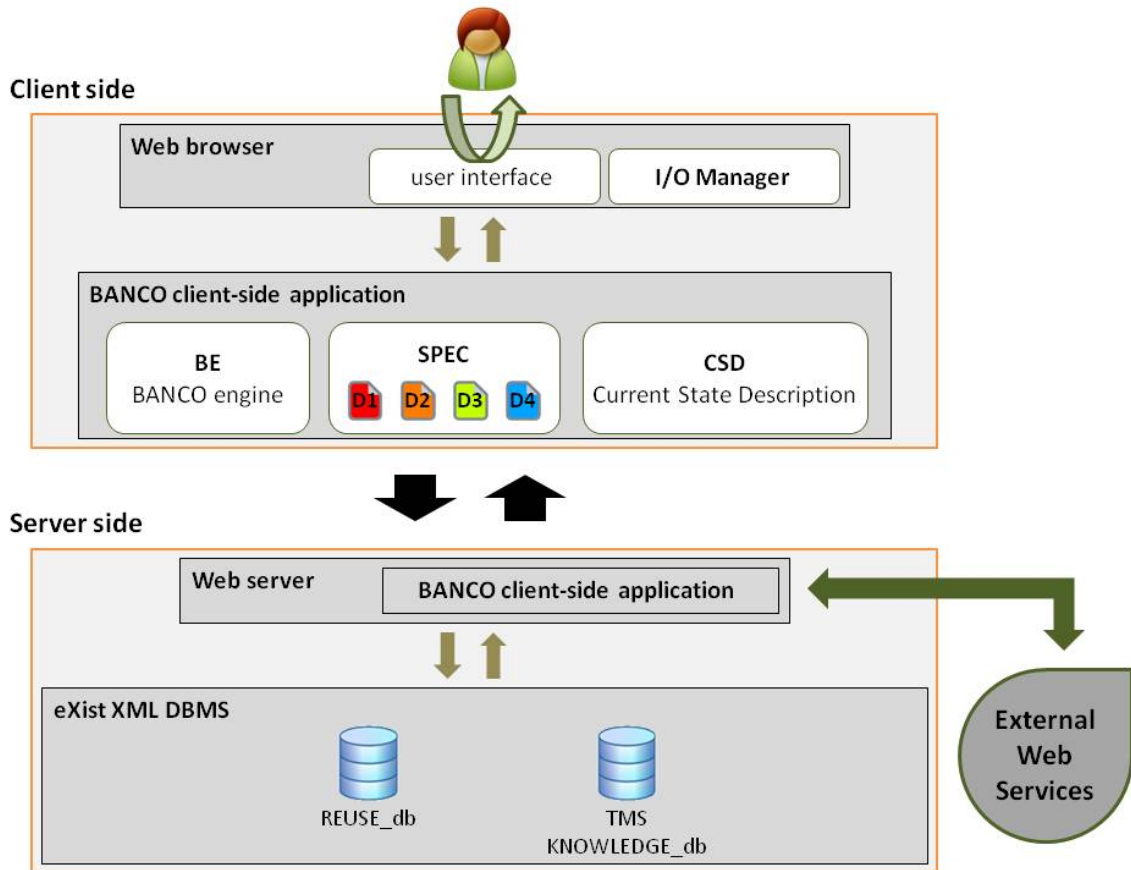


Figure 77. The BANCO architecture implementation for the workflow management case study.

This chapter concludes the last part of this thesis giving an overview of all the implementations of the architecture realized so far. The presentation of the several implementations of the architecture supports its validity, flexibility and feasibility.

Chapter 10:

Usability evaluations for the SCV case study

After having presented the implementations of the BANCO architecture in the five case studies illustrated in Chapter 5, this last chapter of the thesis discusses the results of several usability evaluations performed on the workshops developed in the frame of the SCV (Sistema Culturale Valchiavenna) case study.

The three workshops presented in the previous chapter (Section 9.4) have been evaluated applying different evaluations methods: the workshop for the tourists has been evaluated with a heuristic evaluation, a user test, and two semiotic engineering methods; the workshop for the publisher has been evaluated with a heuristic evaluation; the workshop for the domain experts has been evaluated with a user test. Some of the evaluations have been made in the frame of HCI courses for BSc and MSc classes. The evaluations methods that have been applied aimed at investigating about the interaction problems that the users encounter in using the various workshops. The workshop for the publisher (at meta-design level) has been evaluated with a heuristic evaluation because what was important at that stage was to explore if the workshop was presenting usability problems to be fixed. No experts were involved at that stage of evaluation, just HCI experts. For the domain expert workshop, the user test was chosen as best method of evaluation because the aim was to check the satisfaction of the expert in using the workshop (and in developing a workshop for other people) and to find out if s/he could perform the tasks assigned. For the workshop for the tourists, three different methods have been applied: a) heuristic evaluation, in a

stage in which was important to discover the usability problems without involving the final users but just counting on HCI experts; b) user test, in a further investigation in which final users have been asked to perform some tasks in order to find out interaction problems and to check their satisfaction in using the workshop; c) semiotic engineering methods, in order to check the metacommunication between the software engineers, the HCI experts, the domain experts and the final user (the tourist). These last methods were chosen to prove that the (meta)communication among all the stakeholders in the SSW hierarchy works and supports well the development and end-user development process.

On the basis of the B-architecture, all the workshops have been developed as prototypes following the evolved star life cycle model presented in Chapter 2. Therefore, an user-centered approach has been followed repeating evaluations every time that was needed and choosing the right method at the right time in order to minimize the costs and maximize the results.

10.1. Workshop for tourist

The workshop for tourist has been evaluated following three different evaluation approaches: heuristic evaluation, user test and semiotic engineering methods. The heuristic evaluation has been chosen because was the first attempt to test the usability of the workshop prototype involving HCI experts and not final users. After this first evaluation the problems that emerged have been fixed and a user test has been organized in order to test the workshop prototype with final users, the tourists. A further cycle of evaluation has been performed using two semiotic engineering methods, one predictive method and one user test. These two evaluations have been performed to test the metacommunication between the user and the designer.

10.1.1. Heuristic evaluation

The heuristic evaluation has been performed by 4 evaluators, all HCI experts, and no observers. The heuristics adopted are the Nielsen's ten usability heuristics (Nielsen, 1994):

- 1) **Visibility of system status:** the system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- 2) **Match between system and the real world:** the system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- 3) **User control and freedom:** users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- 4) **Consistency and standards:** users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- 5) **Error prevention:** even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
- 6) **Recognition rather than recall:** minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- 7) **Flexibility and efficiency of use:** accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- 8) **Aesthetic and minimalist design:** dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- 9) **Help users recognize, diagnose, and recover from errors:** error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- 10) **Help and documentation:** even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation.

Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

The 10 Nielsen's heuristic can be grouped on the basis of the class of problems they belong to: perception, cognition, and errors management. The first three heuristics (Visibility of system status, Match between system and the real world, and User control and freedom) belong to the perception class. The next four heuristics (Consistency and standards, Error prevention, Recognition rather than recall, and Flexibility and efficiency of use) belong to the cognition class. Finally the last three heuristics (Aesthetic and minimalist design, Help users recognize, diagnose, and recover from errors, and Help and documentation) belong to the errors management class.

On the basis of this classification, the results of the heuristic evaluation performed on the workshop for tourist can be represented by the graph in Figure 78.

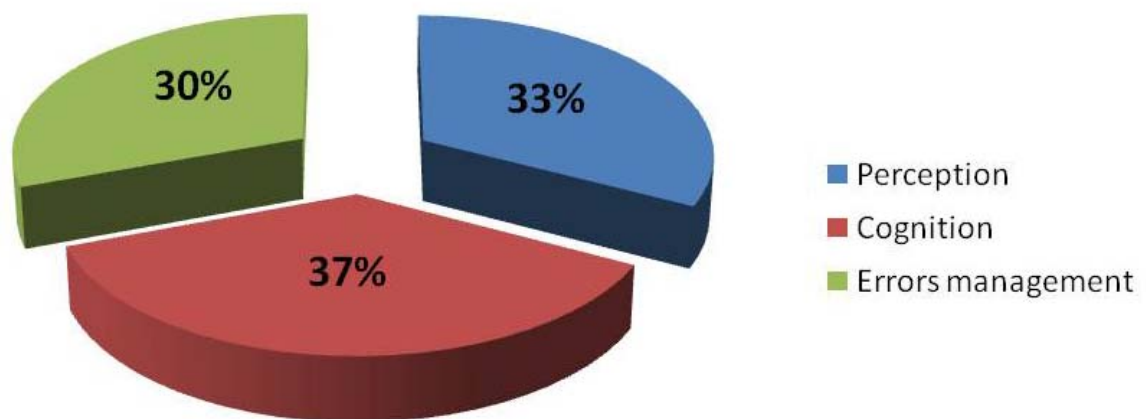


Figure 78. The problems detected in the heuristic usability evaluation performed on the workshop for tourist, classified according to the class of problems they belong to.

Some examples of usability problems detected are the following:

- Perception: the link button to the home page is not visible.
- Cognition: the glossary is reachable only during the annotation creation.
- Errors management: when the maximum level of zoom is reached the zoom reset without displaying any message.

The usability problems detected can be also classified according to another class, that is related to the area of intervention required to fix the problem: graphics, architecture, and programming. According to this classification, the problems detected are divided into three groups illustrated in Figure 79.

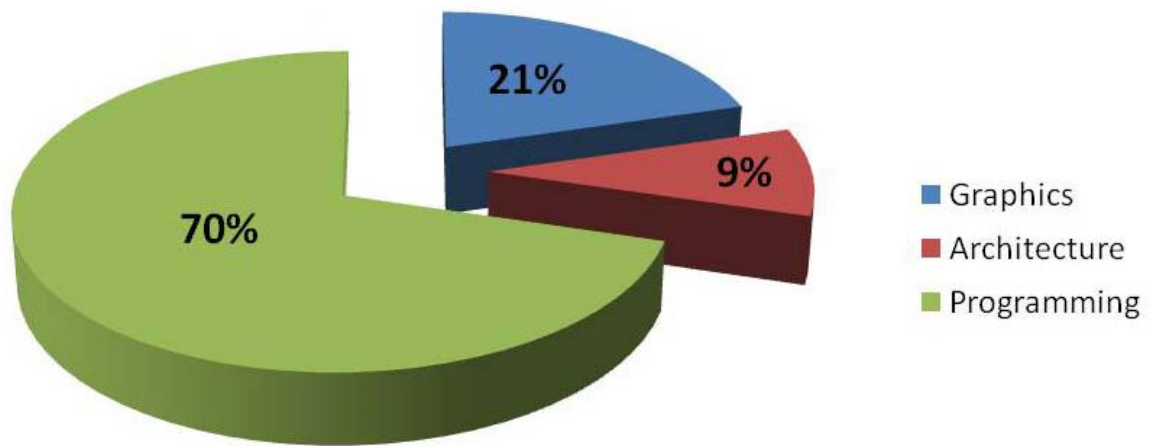


Figure 79. The classification of the problems detected in the heuristic usability evaluation according to the class of area of intervention they belong to.

The usability problems used above as examples may be classified as follows:

- Graphics: the link button to the home page is not visible.
- Architecture: the glossary is reachable only during the annotation creation.
- Programming: when the maximum level of zoom is reached the zoom reset without displaying any message.

What emerges from the heuristic evaluation of the workshop for tourist is that it is needed to redesign its graphics and that more help messages and a user guide have to be added in order to help the user in the performance of her/his activities.

10.1.2. User test

The user test conducted on this workshop prototype followed the within groups modality. Since this modality is affected by the learning effects, the users have been distributed over

three distinct groups. Each group was required to perform a set of tasks. The tasks were the same for all the groups but were listed in different orders.

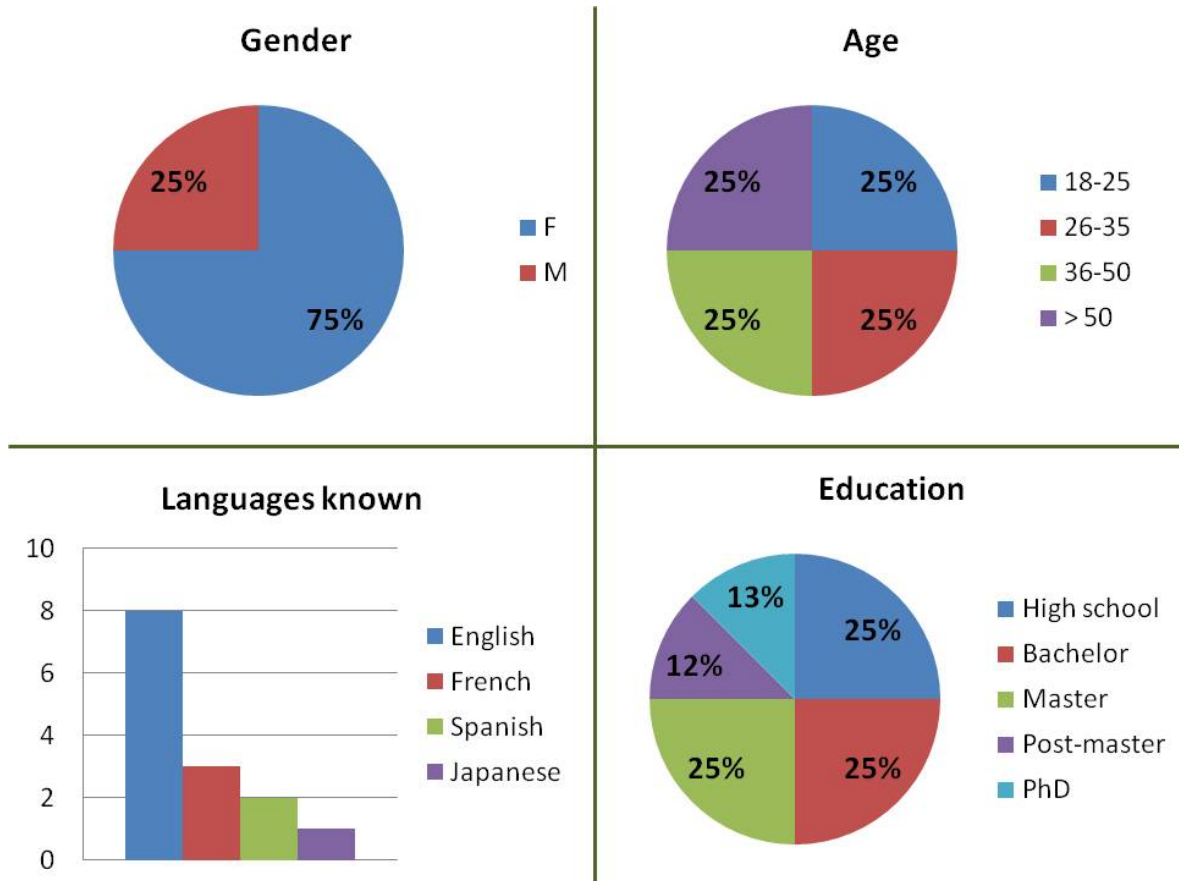


Figure 80. The profile of the users involved in the user test.

Two members of the project team were involved as observers. The think-aloud approach was used. For each task performed by the user, the observer took note of:

- Starting time
- Duration of the execution (if the user does not complete a task in the maximum time allowed, the observer has to ask the user to pass to the next task)
- List of errors committed by the user (if an error is committed more than once, the number of repetitions has to be noted)
- Questions of the user
- Comments of the user on the system

- Satisfaction or frustration expressions
- Usability problems
- Problems in understanding the meaning of a word
- System crash
- Suggestions proposed by the user

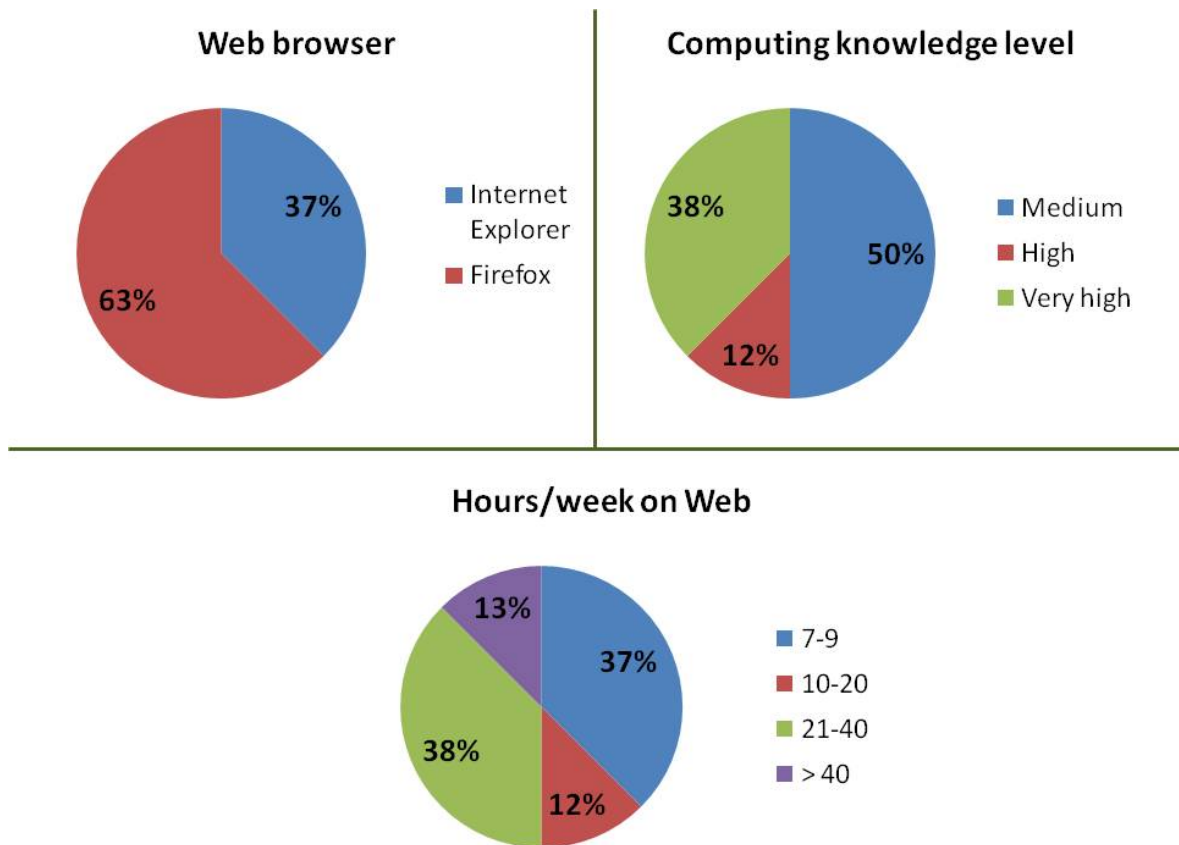


Figure 81. The users' computing knowledge and Web use information emerging from the initial questionnaire.

Eight users were involved in the test. After a brief presentation of the SCV project and the goal of the experiment, the users were asked to fill in an initial questionnaire used to determine the users' profile. The results of the initial questionnaire are illustrated in Figure 80. All the users were of Italian mother tongue.

From the initial questionnaire other information related to the users' profile (see Figure 80), in particular related to their computing knowledge level and their use of the Web (see Figure 81). The computing knowledge level is measured as a) medium, if the user uses the

computer to browse the Web, to send e-mails, to chat with other users and if is able to use text editors and spreadsheets, b) high, if the user uses the computer also to update a blog or a personal website, to write on forums and eventually to create simple macros within text editors and spreadsheets, and c) very high, if the user has programming competences, in one or more programming languages and different paradigms. All the users used Microsoft Windows operating systems. Half of them used interactive systems for annotation of geographical maps on the Web (Google Maps).

Four tasks were assigned to the users:

- Task 0: explore the system
- Task A: read an existing annotation
- Task B: create a new annotation
- Task C: read a narration

The tasks have been assigned to three groups of users that performed them in different orders:

- Group 1: 0, A, B, C
- Group 2: 0, C, A, B
- Group 3: 0, C, B, A

The Group 1 and Group 2 were composed by 3 persons and Group 3 by 2 persons.

Task 0

- 1) Open the Web browser
- 2) Visit the Web page <http://localhost/BancoTerritorio/home.html>
- 3) Log in with username= “visitor” and password = “visitor”
- 4) Explore the Web page for 3 minutes
- 5) Log out

Task A

- 1) Visit the Web page <http://localhost/BancoTerritorio/home.html>
- 2) Log in with username= “visitor” and password = “visitor”
- 3) Open the map of Valchiavenna
- 4) Open the annotation associated to a disappointed emoticon
- 5) Close the annotation
- 6) Close the map
- 7) Log out

Task B

- 1) Visit the Web page <http://localhost/BancoTerritorio/home.html>
- 2) Log in with username= “visitor” and password = “visitor”
- 3) Open the map of Chiavenna
- 4) Choose the emoticon associated to appreciation and use it to annotate the map
- 5) Put as title “Fontana del 1700”
- 6) Put as text of the annotation “Interessante fontana del 1700”
- 7) Save the annotation
- 8) Close the map
- 9) Log out

Task C

- 1) Visit the Web page <http://localhost/BancoTerritorio/home.html>
- 2) Log in with username= “visitor” and password = “visitor”
- 3) Open the map of Chiavenna
- 4) Open an existing narration
- 5) Close the narration
- 6) Close the map

7) Log out

Figure 82 shows the time of execution (in seconds) of the various tasks performed by the different groups.

GROUP 1	User 1	User 2	User 3	Average
Task A	62	59	110	77,00
Task B	---	---	---	---
Task C	42	50	173	88,33

GROUP 2	User 1	User 2	User 3	Average
Task C	75	---	118	96,50
Task A	51	150	---	100,50
Task B	108	---	138	123,00

GROUP 3	User 1	User 2	Average
Task C	78	110	94
Task B	---	---	---
Task A	65	51	58

Figure 82. The tasks execution times and averages of each user in the three groups.

Where the time is not present (“---“), the tasks were not completed by the users. Task A was not completed by 1 user, as well as Task C, while Task B was not completed by 6 users.

Figure 83 resumes the averages of the execution times of the three groups of users and calculates for each task the global average of execution time.

The results that emerge from the user test on the workshop for tourist and from the final questionnaire that the users were asked to fill in, coincide with the ones emerged from the

heuristic analysis: the workshop for tourist is affected by problems related to the graphical interface that needs to be redesigned.

	Group 1	Group 2	Group 3	Average
Task A	77,00	100,50	58,00	78,50
Task B	---	123,00	---	123,00
Task C	88,33	96,50	94,00	92,94

Figure 83. The tasks execution times and averages of each group.

The problems related to the graphical interface are highlighted by the high execution times related to task B that in 2 groups was even not performed at all. Moreover, it emerges that a user guide is needed to help the user in understanding the tools available in the workshop.

10.1.3. Semiotic engineering evaluations

Two methods of semiotic engineering evaluation have been used to evaluate the metacommunication between the user of the workshop for the tourist and the workshop designer: the semiotic inspection method (SIM) and the communicability evaluation method (CEM). The two methods have been defined in (de Souza et al., 2006; Prates et al., 2000a; Prates et al., 2000b; de Souza, 2005; Sharp, Rogers & Preece, 2007; de Souza and Leitão, 2009) as applications of semiotic engineering theory to support professional HCI activities.

SIM method explores the emission of the metacommunication, trying to reconstruct the messages sent by the designer to the targeted users.

CEM method explores the reception of the metacommunication, trying to identify through users observation the empirical evidence of the effects that the designer's messages have on the users' interaction.

As to the SIM method applied to the workshop for the tourist, the following scenario was decided and used by three evaluators in the exploration of the system:

The user is very keen on trekking and, in particular, s/he knows very well the Valchiavenna valley. Her/his friend, an expert in computer science, suggests her/him to visit a Web portal that allows to browse Valchiavenna maps and to annotate them leaving comments on it. The user is very interested in visiting the Web portal because s/he wants to read the comments left by other persons and wants eventually leave some comments too. Therefore, the user accesses to the Web portal and after selecting the map of interest s/he discovers to have all the tools needed to browse the map and to annotate it. After having read some comments created by some students, the user decides to annotate a specific location (Madesimo), a town in Valchiavenna valley. Therefore, s/he zooms in the map and leaves an annotation related to an “appreciation” emoticon.

The SIM evaluation leads to the conclusion that the workshop is simple because the designer wanted to keep it as simple as possible, but the tools available are sufficient for the navigation and annotation of the geographical maps. A problem detected regards the fact that very often static signs are used instead of metalinguistic signs that are used just for presenting error or warning messages but not user guides or helps.

As to the CEM evaluation, a group of six users have been involved in the test. Two evaluators have been involved as observers and were in charge of recording the test and of taking note of the communication breakdowns detected.

After the user test, the evaluators performed what is called the tagging of the communication breakdowns identified. At each breakdown, one or more of the 13 semiotic tags defined in (de Souza and Leitão, 2009) were assigned:

- 1) “I give up.”
- 2) “Looks fine to me.”

- 3) “Thanks, but no, thanks.”
- 4) “I can do otherwise.”
- 5) “Where is it?”
- 6) “What happened?”
- 7) “What now?”
- 8) “Where am I?”
- 9) “Oops!”
- 10) “I can’t do it this way.”
- 11) “What is this?”
- 12) “Help!”
- 13) “Why doesn’t it?”

These tags represent utterances that characterize communicative breakdowns between the user and the designer’s deputy, that is the workshop under evaluation.

The frequency and patterns of presentation of the tags are illustrated in Figure 84.

From a post-test questionnaire submitted to the users, emerges that they appreciate the goal of the workshop but not the graphical interface that they consider too simple and not in line with the more used interactive systems for maps annotation and browsing (e.g. Google Maps). However, the use of the emoticons to assign a mood to a particular point of interest on the map is very appreciated by the users and the meaning of the emoticons as visual links has been recognized easily.

After the application of both the semiotic engineering methods of evaluation, the semiotic profile that emerged was describing a user who is interested in the exploration and annotation of geographical maps, who has a medium knowledge of the use of Web and related technologies (s/he can use a Web browser and knows how to interact with a dynamic Web site). What emerges from the two evaluations is that the workshop for tourist needs to be graphically redesigned and that a user guide should be added in order to help the users in understanding how to use the tools that are available in the workshop.

TAG	FREQUENCY
Why doesn't it?	4
Where is it?	3
I give up.	3
What happened?	3
I can't do it this way.	2
What now?	2
Looks fine to me.	1

PATTERN	FREQUENCY
Why doesn't it? + I give up.	2
Where is it? + I can't do it this way.	1
What happened? + What now?	1
Where is it? + I give up.	1

Figure 84. The frequency of tags and patterns in the CEM evaluation of the workshop for tourists.

10.2. Workshop for publisher

For the workshop for publisher the method of evaluation chosen was the heuristic evaluation because this workshop prototype is at an early stage of development and this evaluation method allows to test the usability involving HCI experts and not the final users, keeping low the costs for the analysis.

10.2.1. Heuristic evaluation

The heuristic evaluation has been performed by six evaluators, all HCI experts, and no observers. The heuristics adopted are the Nielsen's ten usability heuristics (Nielsen, 1994).

On the basis of the classification on classes of problems, the results of the heuristic evaluation are represented in Figure 85.

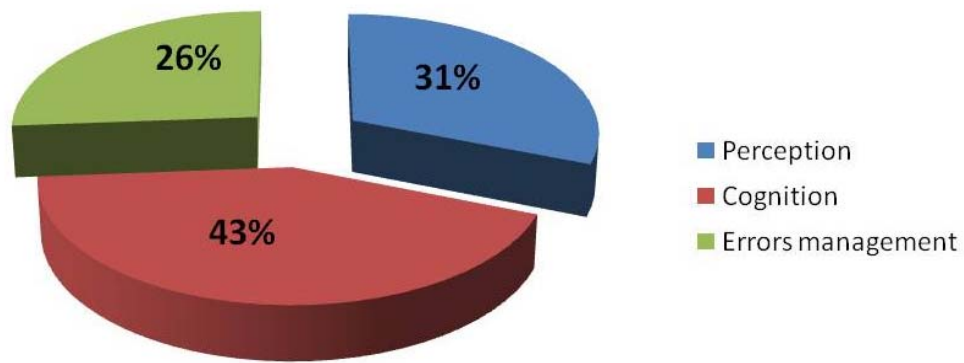


Figure 85. The problems detected in the heuristic usability evaluation performed on the workshop for publisher, classified according to the class of problems they belong to.

Some examples of usability problems detected in the heuristic analysis on this workshop are the following:

- Perception: when the mouse moves over active areas of the interface, no tooltips appear.
- Cognition: the menu for map selection slides down without stopping when the list of available maps ends.
- Errors management: once an annotation is created, its visual link cannot be moved to another position.

The usability problems detected on this workshop can be also classified according to the area of intervention required to fix them. This classification is shown in Figure 86.

The usability problems used above as examples may be listed according to this classification as follows:

- Graphics: when the mouse moves over active areas of the interface, no tooltips appear.
- Architecture: the menu for map selection slides down without stopping when the list of available maps ends.
- Programming: once an annotation is created, its visual link cannot be moved to another position.

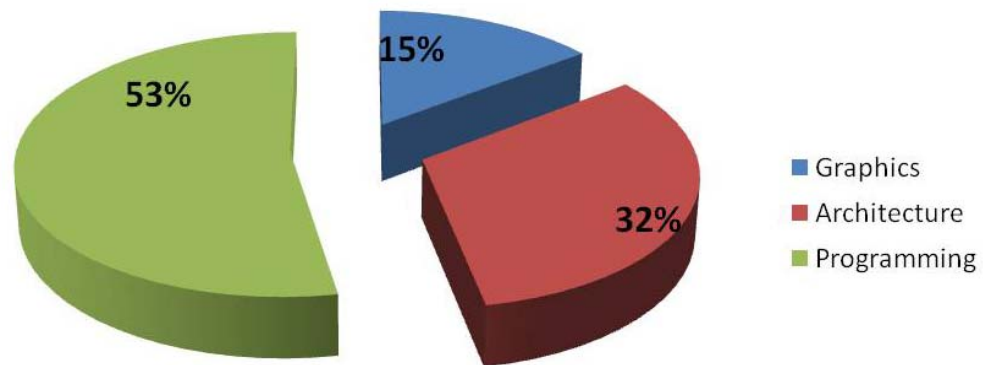


Figure 86. The classification of the usability problems of the workshop for publisher according to the class of area of intervention.

The same observations emerged for the workshop for tourist are valid in this case: the graphic interface is poor and needs to be redesigned. Also for this workshop, more help messages and a user guide are needed in order to guide the publisher in the performance of her/his activities.

10.3. Workshop for domain expert

For this workshop the method of evaluation used was chosen because it constitutes the implementation of the architecture that better highlight the importance of end-user development activities. Therefore, some domain experts were involved as users in order to test the validity of the workshop prototype developed.

10.3.1. User test

The user test on the workshop for domain expert was performed involving five experts of different domains: an archeologist, a geographer, a geologist, a researcher in computer science applied to tourism and cultural heritage, and a researcher and developer in tourism and cultural heritage field.

Each user was required to perform a set of tasks. The tasks were the same for all the groups listed in the same order.

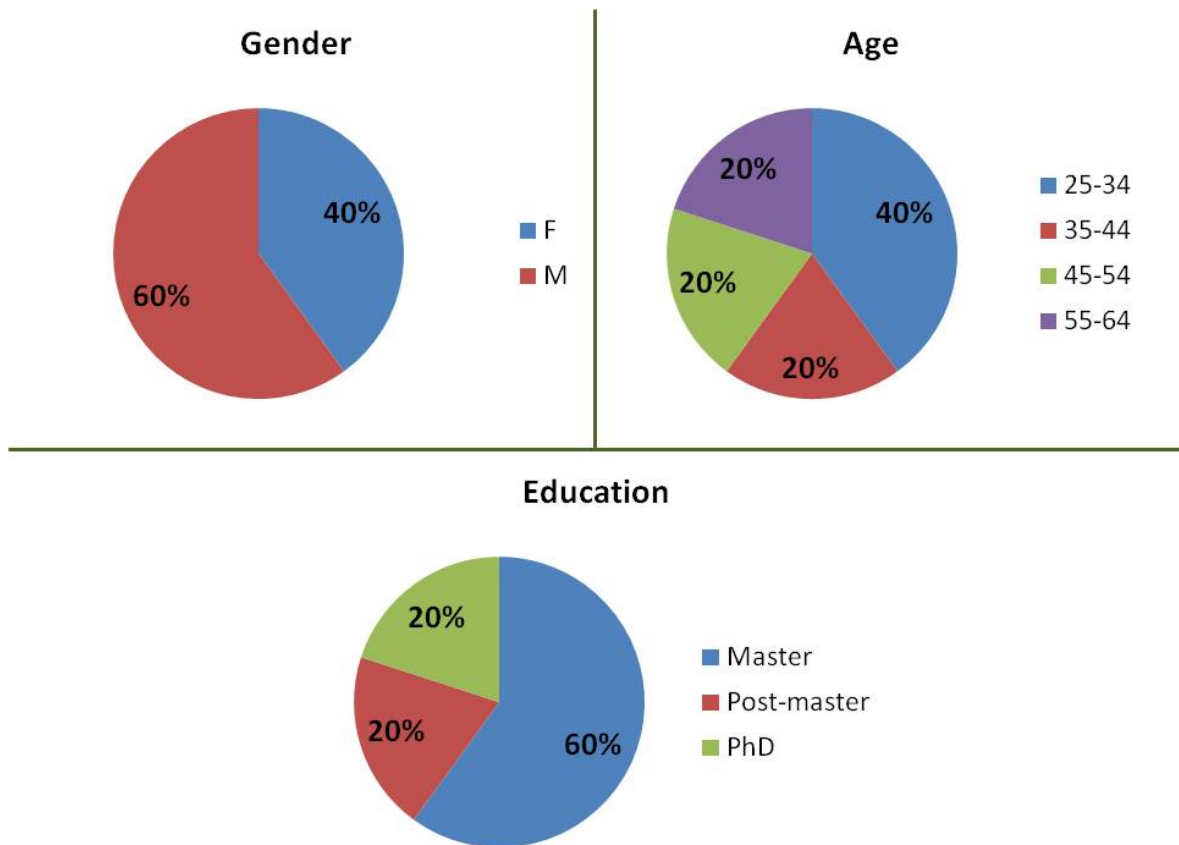


Figure 87. Gender, age, and education level for the group of users involved in the user test for the workshop for the domain expert.

The think-aloud approach was used. For each task performed by the user, one observer took note of:

- Starting time
- Duration of the execution (if the user does not complete a task in the maximum time allowed, the observer has to ask the user to pass to the next task)
- List of errors committed by the user (if a error is committed more than once, the number of repetitions has to be noted)
- Questions of the user
- Comments of the user on the system
- Satisfaction or frustration expressions
- Usability problems
- Problems in understanding the meaning of a word

- System crash
- Suggestions proposed by the user

After a brief presentation of the SCV project and the goal of the experiment, the users were asked to watch a video showing the interaction of a tourist with her workshop.

After that the users were asked to fill in an initial questionnaire used to determine the users profile. The results of the initial questionnaire are illustrated in Figure 87, Figure 88, and Figure 89. All the users were of Italian mother tongue and use the computer both for work and for personal use.

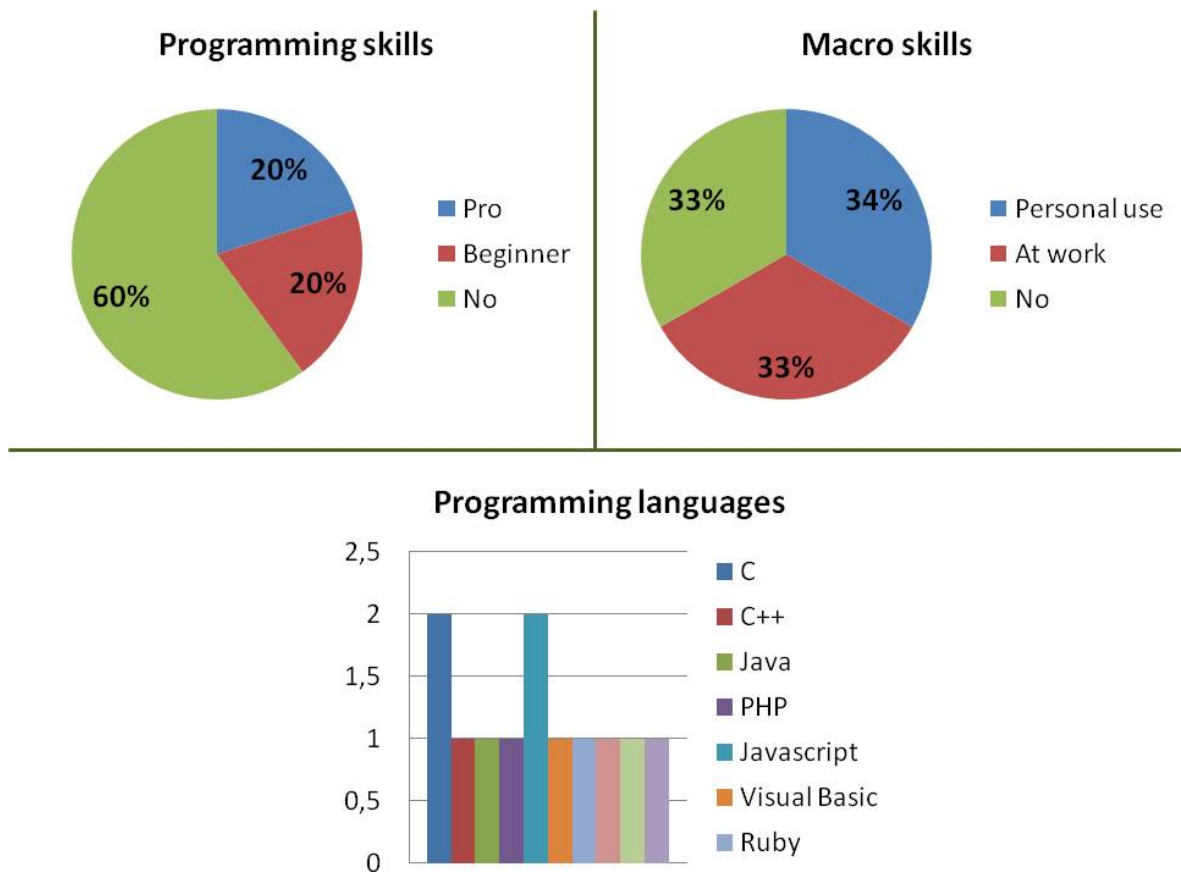


Figure 88. Programming and macro skills, and programming languages used by the domain experts involved in the user test.

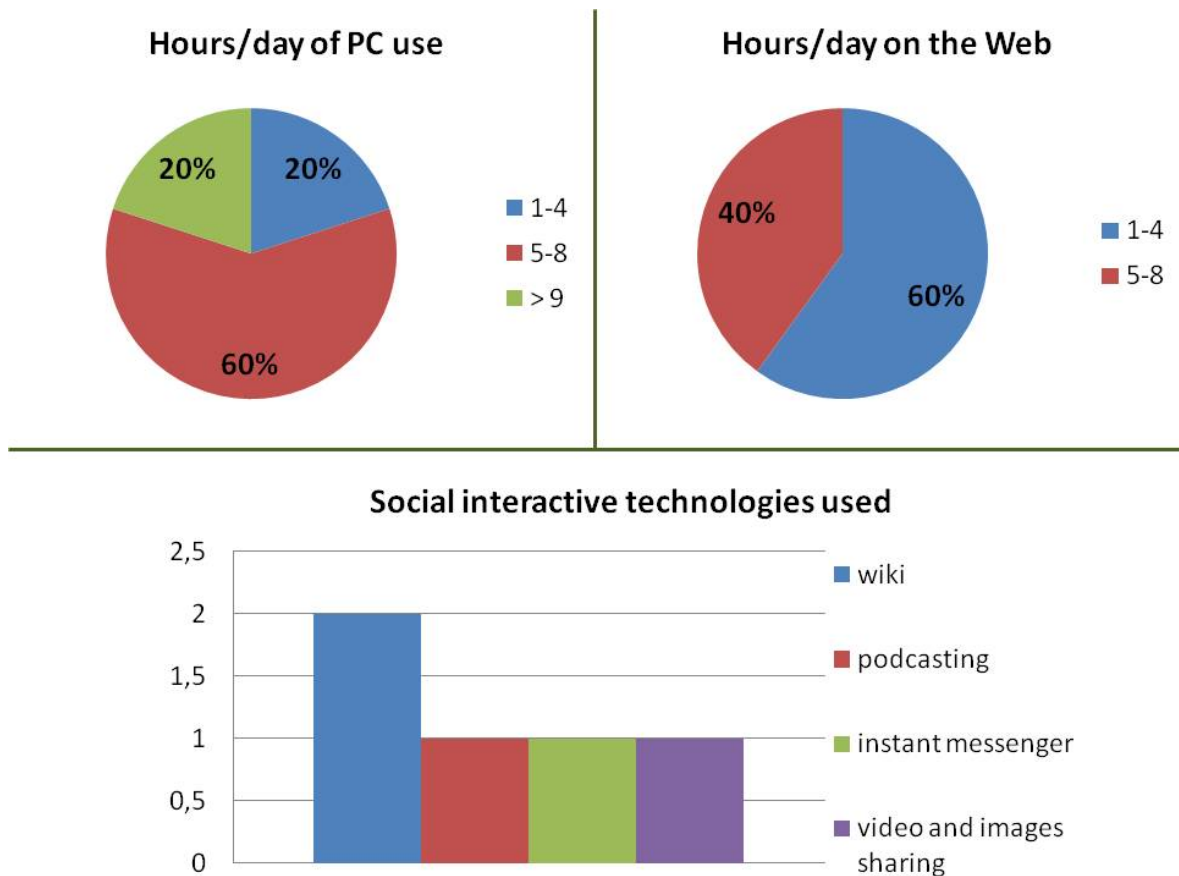


Figure 89. Data about the use of the computer by the domain expert. Time spent on the PC, on the Web and social interactive technologies used by them.

Eleven tasks were assigned to the users:

- Task 1: access the workshop for domain expert
- Task 2: open the user guide and read the content
- Task 3: add a workbench to the current project
- Task 4: add the name of the map to the workbench
- Task 5: add the box for the window management buttons
- Task 6: add a set of buttons to that box
- Task 7: add the box for the annotation management buttons
- Task 8: add a set of buttons to that box
- Task 9: remove from the window management buttons box all the buttons except the one that closes the window
- Task 10: save the workbench

- Task 11: log out

Figure 90 shows the time of execution (in seconds) of the various tasks performed by the different domain experts involved in the user test. All the users perform every task.

	User 1	User 2	User 3	User 4	User 5	Average
Task 1	7	5	10	10	5	7,4
Task 2	5	4	9	6	3	5,4
Task 3	6	2	3	19	2	6,4
Task 4	2	1	4	5	2	2,8
Task 5	2	1	5	4	3	3
Task 6	8	6	15	10	11	10
Task 7	2	1	2	4	3	2,4
Task 8	10	10	10	14	13	11,4
Task 9	15	20	20	20	14	17,8
Task 10	2	2	2	3	2	2,2
Task 11	6	2	2	3	3	3,2

Figure 90. The tasks execution times and averages of each user involved in the user test.

A final questionnaire has been submitted to the domain expert at the end of the user test. It was composed by three sets of questions:

- 1) A first set is a SUS (system usability scale) questionnaire, a ten-item attitude Likert scale (Brooke, 1996).
- 2) The other set is a CUSQ (computer usability satisfaction Questionnaires) questionnaire, a nineteen-item attitude Likert scale (Lewis, 1993).
- 3) A last questionnaire, a six-item attitude Likert scale chosen on the basis of the domain of the case study (i.e. tourism and cultural heritage).

What emerges from the user test performed on the workshop for domain expert, and from the final questionnaire that the users were asked to fill in, is that they strongly agree on the importance of tools that allow end users to become developers of their environment or of environments for other people. They all appreciate the simplicity of the workshop but, also in this case, the graphic interface was considered not in line with the most well known Web application currently used.

This chapter concludes the last part of this thesis presenting the results of the evaluation of the workshops developed for the SCV project using the architecture presented in Chapter 8. All the evaluations performed show the feasibility of the implementation of the architecture to support a network of communities of practice collaborating to a common collaborative project. However, the results suggest that some aspects of the workshop have to be improved, like the help and user guide features and their look and feel to make them more appealing.

Conclusions and future developments

The research context in which this thesis is grounded is end-user development. In particular, the thesis aims at offering models and tools to enable end-user development supporting the activity of global communities. This specific context is affected by several challenges that depend on the multiculturalism of the stakeholders who participate and collaborate in a design process becoming end-user developers.

This thesis presents a semiotic model for end-user development (Part III) and a Web architecture, the BANCO architecture, that supports the development of software systems (Part IV). The BANCO architecture adopts the semiotic model and the mediation mechanisms presented in the thesis. The combination of the mediation mechanisms, the semiotic model, and the architecture face the challenges that emerge from the research context described in the Part I.

The software systems developed using the BANCO architecture a) are localized to the end users' culture, role played in the context, and digital platform in use, b) support the co-evolution phenomenon by which the users evolve in using a system and the system should be evolved according to the evolved users' needs, and c) permits the communication among all the stakeholders involved in the same collaborative project.

Part II and Part V present five case studies and the implementation of the architecture for them. The feasibility, flexibility and validity of the implementation of the architecture is proved by the description of its application in the several case studies and by the presentation of the results of several usability evaluations.

Among the case studies presented, two of them are still under development: the Sistema Culturale Valchiavenna project (tourism and cultural heritage context) and the Task Management System project (workflow management context).

As to Sistema Culturale Valchiavenna project, the evaluations of the workshops developed for the tourists, the publishers, and the domain experts, suggest the redesign of their graphics. The look and feel, as pointed out by the users and the evaluators involved in the evaluations, should be changed according to the look and feel of the currently most used applications for collaborative activities on the Web. Furthermore, more features related to help and user guides should be added to the workshops. This means that, according to the star life cycle presented in Chapter 2, a new phase of development has to be planned, followed by another session of evaluation. However, the results obtained by the involvement of several persons in the evaluations of the workshops confirm the interest that end users have in performing end-user development activities.

As to the Task Management System project, a first prototype is under development and as soon as it will be ready, a first evaluation session will be planned. The Task Management System project constitutes an important step in the extension of the BANCO architecture, with the use of external Web service. This extension grants to the architecture a higher interoperability with other systems and supports even more the collaborative work settings.

The semiotic model for end-user development and the mediation mechanisms presented in this thesis are going to be applied in conjunction with the work developed for the PhD project of Li Zhu at Università degli Studi di Milano (Zhu, 2010). This project is focused on the definition of the Hive-Mind Space Model (HMS), a framework to support meta-design and end-user development in global communities through the use of boundary objects and boundary zones.

Moreover, an extension of the semiotic engineering methods is under study. The semiotic inspection method (SIM) and the communicability evaluation method (CEM), used in this thesis for the evaluation of the workshop for the tourist, analyze the metacommunication between the user and the designer focusing on static, dynamic and metacommunication signs. With the extension of the methods, some other aspects would be added to their

definition in order to investigate on the influence that creative design techniques have on metacommunication.

References

- Amaya (n.d.). Amaya Web Editor. Retrieved from <http://www.w3.org/Amaya/>
- Andersen, P. B. (1992). Computer semiotics. *Scandinavian Journal of Information systems*, 4, 3-30.
- Andersen, P. B., Holmqvist, B., & Jensen, J. F. (Eds.). (1993). *The computer as medium*. Cambridge, MA: Cambridge University Press.
- Andersen, P. B. (1997). *A theory of computer semiotic: semiotic approaches to construction and assessment of computer systems (2nd edition)*. Cambridge, MA: Cambridge University Press.
- Andersen, R., & Mørch, A. I. (2009). Mutual Development: A Case Study in Customer-Initiated Software Product Development. In V. Pipek, M. B. Rosson, B. Ruyter, & V. Wulf (Eds.), *Proc. of IS-EUD2009* (pp. 31-49). Berlin, Germany: Springer-Verlag.
- Annotea Project (n.d.). Retrieved from <http://www.w3.org/2001/Annotea>
- Annozilla (n.d.). Annozilla (Annotea on Mozilla). Retrieved from <http://annozilla.mozdev.org/>
- Arias, E., Eden, H., Fischer, G., Gorman, A., & Scharff, E. (2000). Transcending the individual human mind - creating shared understanding through collaborative design. *ACM TOCHI*, 7(1), 84-113.

- Aroni, S., Baroni, P., Fogli, D., & Mussio, P. (2002): Supporting coevolution of users and systems by the recognition of Interaction Patterns. In *Proc. of AVI 2002* (pp. 177-189). New York, NY: ACM Press.
- Åsand, H., & Mørch, A. (2006). Super Users and Local Developers: the Organization of End User Development in accounting company. *Journal of Organizational and End User Computing*, 18(4), 1-21.
- Baecker, R. M. (1995). *Readings in human-computer interaction: toward the year 2000*. San Francisco, CA: Morgan Kaufmann Publishers.
- Baldwin, C.Y., & Clark, K.B. (2000). *Design Rules: The Power of Modularity*. Cambridge, MA: The MIT Press.
- Barber, W., & Badre, A. N. (1998). Culturability: The Merging of Culture and Usability. In *Proc. of Human Factors and the Web*. Retrieved from <http://www.research.att.com/conf/hfweb/proceedings/barber/index.htm>
- Barricelli, B. R., Iacob, C., & Zhu, L. (2009). Map-Based Wikis as Contextual and Cultural Mediators. Presented at *Community Practices and Locative Media workshop at Mobile HCI09*. Available online: <http://www.uni-siegen.de/locatingmedia/workshops/mobilehci/>
- Barricelli, B. R., Iacob, C., & Zhu, L. (2010). BANCO Web Architecture to Support Global Collaborative Interaction Design. In V. Evers, J. Abdelnour-Nocera, & E., del Galdo (Eds.), *Proc. of IWIPS 2010* (pp. 159-162). London, UK: P&SI.
- Barricelli, B. R., Marcante, A., Mussio, P., Parasiliti Provenza, L., Padula, M., & Scala, P. L. (2009). Designing Pervasive and Multimodal Interactive Systems: An Approach Built on the Field. In P. Grifoni (Ed.), *Handbook of Research on Multimodal Human Computer Interaction and Pervasive Services: Evolutionary Techniques for Improving Accessibility* (pp. 243-264). Hershey, PA: IGI Global Publication.
- Barricelli, B. R., Marcante, A., Mussio, P., Parasiliti Provenza, L., Valtolina, S., & Fresta, G. (2009). BANCO: a Web Architecture Supporting Unwitting End-User Development. *IxD&A*, 5/6, 23-30.

- Barricelli, B. R., Mussio, P., Padula, M., Piccinno, A., Scala, P. L., & Valtolina, S. (2010). Visual Workflow Composition through Semantic Orchestration of Web Services. Presented at *EUD4Services Workshop*. Available online: <http://cslab.dico.unimi.it/EUD4Services/>
- Barricelli, B. R., Mussio, P., Padula, M., Piccinno, A., Scala, P. L., & Valtolina, S. (2010). Interactive Task Management System Development Based on Semantic Orchestration of Web Services. Presented at *ItAIS 2010*. Available online: <http://www.cersi.it/itais2010>
- Barricelli, B. R., Mussio, P., Padula, M., & Scala, P. L. (2010). TMS for Multimodal Information Processing. Will appear in *Multimedia Tools and Applications, Springer*. Online first: <http://dx.doi.org/10.1007/s11042-010-0527-x>
- Barricelli, B. R., Padula, M., & Scala, P. L. (2009). TMS for Multimodal Information Processing. In A. M. Tjoa & R. R. Wagner (Eds.), *Proc. of DEXA2009* (pp. 295-299). Los Alamitos, CA: IEEE Computer Society.
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 5, 34-43.
- Bianchi, A., Bottoni, P., & Mussio, P. (1999). Issues in Design and Implementation of Multimedia Software Systems. In *Proc. of ICMCS'99* (pp. 91-96). Los Alamitos, CA: IEEE Computer Society.
- Bødker, S., Ehn, P., Knudsen, J., Kyng, M., & Madsen, K. (1988). Computer support for cooperative design. In *Proc. of CSCW '88* (377-394). New York, NY: ACM Press.
- Bødker, K., & Pedersen, J. (1991). Workplace Cultures: Looking at Artifacts, Symbols and Practices. In J. Greenbaum, & M. Kyng M. (Eds.), *Design at Work: Cooperative Design of Computer Systems* (pp. 121-136). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Bødker, S., & Gronbaek, K. (1991). Design in action: From prototyping by demonstration to cooperative prototyping. In J. Greenbaum, & M. Kyng (Eds.), *Design at work: Cooperative design of computer systems* (pp. 197-218). Hillsdale, NJ: Lawrence Erlbaum Associates.

- Bødker, S., Gronbaek, K., & Kyng, M. (1993). Cooperative Design: Techniques and Experiences From the Scandinavian Scene. In D. Schuler, & A. Namioka (Eds.), *Participatory Design: Principles and Practices* (pp. 157-175.). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Borchers, J. (2001). *A Pattern Approach to Interaction Design*. Hoboken, NJ: John Wiley & Sons.
- Borchers, J., Fincher, S., Griffiths, R., Pemberton, L., & Siemon, E.(2001). Usability Pattern Language: Creating A Community. *Journal of Human-Centred Systems and Machine Intelligence*, 15(4), 377-385.
- Borgman, C. L. (1992). Cultural diversity in interface design. *SIGCHI Bull.*, 24(4), 31.
- Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E., & Trinchese, R. (2004). MADCOW: a multimedia digital annotation system. In M. F. Costabile (Ed.), *Proc. of AVI 2004* (pp. 55-62). New York, NY: ACM Press.
- Bottoni, P., Costabile, M. F., & Mussio, P. (1999). Specification and Dialogue Control of Visual Interaction through Visual Rewriting Systems. *ACM TOPLAS*, 21(6), 1077-1136.
- Boulle, L. (2005). *Mediation: Principles, Process, Practice*. London, UK: LexisNexis Butterworths.
- Bourguin, G., Derycke, A., & Tarby, J. C. (2001). Beyond the Interface: Co-evolution inside Interactive Systems - A Proposal Founded on Activity Theory. In A. Blandford, J. Vanderdonckt, & P. Gray (Eds.), *Proc. of IHM-HCI 2001* (297-310). Toulouse, France: Cépaduès-Éditions.
- BPEL4WS (n.d.). Web Services Business Process Execution Language: OASIS Standard. Retrieved from <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- Brancheau, J. C., & Brown, C. V. (1993). The Management of End-User Computing: Status and Direction. *ACM Computing Surveys*, 25(5), 437-482.
- Branham, S., Golovchinsky, G., Carter, S., & Biehl, H. T. (2010). Let's Go from the Whiteboard: Supporting Transitions in Work through Whiteboard Capture and Reuse. In *Proc. of CHI'10* (pp. 75-84). New York, NY: ACM Press.

- Brodie, C. B., & Hayes, C. C. (2002). DAISY: A Decision Support Design Methodology for Complex, Experience-Centered Domains. *IEEE TSMCA*, 32(1), 50-71.
- Brooke, J. (1996). SUS: a "quick and dirty" usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.), *Usability Evaluation in Industry* (pp. 189-194). London, UK: Taylor and Francis.
- Brown, J. S., & Duguid, P. (2000). *The social life of information*. Boston, MA: Harvard Business School Press.
- Bruns, A. (2008). *Blogs, Wikipedia, Second Life, and Beyond: from production to produsage*. New York, NY: Peter Lang.
- Cadieux, P., & Esselink, B. (2002). GILT: Globalization, Internationalization, Localization, Translation. *LISA Globalization Insider*, 1(5). Retrieved from http://www.lisa.org/globalizationinsider/2002/03/gilt_globalizat.html
- Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A Unifying Reference Framework for Multi-target User Interfaces. *Interacting with Computers*, 15(3), 289-308.
- Carmien, S., Dawe, M., Fischer, G., Gorman, A., Kintsch, A., & Sullivan, J. F. (2005). Sociotechnical environments supporting people with cognitive disabilities using public transportation. *ACM TOCHI*, 12(2), 233-262.
- Carrara, P., Fresta, G., & Rampini, A. (2000). Interfaces for Geographic Applications on the World Wide Web: an Adaptive Computational Hypermedia. In P. L. Emiliani, & C. Stephanidis (Eds.), *Proc. of ERCIM Workshop* (pp. 341-342).
- Carstensen P., & Schmidt K. (1999). Computer supported cooperative work: New challenges to systems design. In K. Itoh K. (Ed.) *Handbook of human factors* (pp. 619-636). Tokyo, Japan: Asakura Publishing.
- Carroll, J. M., & Rosson, M. B. (1992). Deliberated Evolution: Stalking the View Matcher in design space. *Human-Computer Interaction*, 6(3-4), 281-318.
- Chaudhri, A. B., Rashid, A., & Zicari, R. (2003). *XML data management: Native XML and XML-enabled database systems*. Boston, MA: Addison-Wesley.

- Cook, B. L. (1980). Picture communication in Papua New Guinea. *Educational Broadcasting International*, 13(2), 78-83.
- Costabile, M. F. (2001). Usability in the software life cycle. In S. K. Chang (Ed.), *Handbook of Software Engineering & Knowledge Engineering. Vol. I* (pp. 179-192). London, UK: World Scientific.
- Costabile, M. F., Fogli, D., Lanzilotti, R., Mussio, P., Parasiliti Provenza, L., & Piccinno, A. (2007). Advancing End-User Development Through Meta-Design. In S. Clarke (Ed.), *End User Computing Challenges Technologies: Emerging Tools and Applications* (pp. 143-167). Hershey, PA: Information Science Reference.
- Costabile, M. F., Fogli, D., Letondal, C., Mussio, P., & Piccinno, A. (2003) Domain-Expert Users and their Needs of Software Development. In C. Stephanidis (Ed.), *Proc. of UAHCI 2003*, (pp. 532-536). Mahwah, NJ: Lawrence Erlbaum Associates.
- Costabile, M.F., Fogli, D., Marcante, A., & Piccinno, A. (2006). Supporting Interaction and Co-evolution of Users and Systems. In A. Celentano (Ed.), *Proc. of AVI 2006* (pp.143-150). New York, NY: ACM Press.
- Costabile, M. F., Fogli, D., Mussio, P., & Piccinno, A. (2006). End-User Development: the Software Shaping Workshop Approach. In H. Lieberman, F. Paternò, & V. Wulf (Eds.), *End-User Development* (pp. 183-205). Dordrecht, The Netherlands: Springer.
- Costabile, M. F., Fogli, D., Mussio, P., & Piccinno, A. (2007). Visual Interactive Systems for End-User Development: a Model-based Design Methodology. *IEEE TSMCA*, 37(6), 1029-1046.
- Costabile, M. F., Fogli, D., Marcante, A., Mussio, P., Parasiliti Provenza, L. and Piccinno, A. (2008). Designing Customized and Tailorable Visual Interactive Systems. *IJSEKE*, 18(3), 305-325.
- Costabile, M., Mussio, P., Parasiliti Provenza, L., & Piccinno, A. (2008). End users as unwitting software developers. In R. Abraham, M. Burnett, & M. Shaw (Eds.), *Proc. of WEUSE '08* (pp. 6-10). New York, NY: ACM Press.

- Costabile, M. F., Mussio, P., Parasiliti Provenza, L., & Piccinno, A. (2009). Supporting End Users to be Co-designers of their Tools. In V. Pipek, M. B. Rosson, B. Ruyter, & V. Wulf (Eds.), *Proc. of IS-EUD2009* (pp. 70-85). Berlin, Germany: Springer-Verlag.
- Courtney, A. J. (1986). Chinese Population Stereotypes: Color Association. *Human Factors*, 28(1), 97-99.
- Cypher, A. (1993). *Watch What I Do: Programming by Demonstration*. Cambridge, MA: The MIT Press.
- d'Amato, R. (2009). *Analisi dei metodi di end-user development: il refactoring dell'architettura BANCO*. M.Sc. thesis. Advisor: Piero Mussio. Co-Advisors: Barbara Rita Barricelli, Loredana Parasiliti Provenza.
- Dale, H. (2004). Emoticon-interpreters create risks in instant messaging services. *The Risk Digest*, 23(48). Retrieved from <http://catless.ncl.ac.uk/Risks/23.48.html#subj5>
- de Mauro, T. (1982). *Minisemantica dei linguaggi non verbali e delle lingue*. Roma, Italia: Laterza.
- De Silva, B., Ginige, A., Bajaj, S., Ekanayake, A., Shirodkar, R., & Santa, M. (2009). A Tool to Support End-User Development of Web Applications Based on a Use Case Model. In *Proc. of ICWE 2009* (pp. 527-530).
- de Souza, C. S. (1993). The semiotic engineering of user interface languages. *International Journal of Man-Machine Studies*, 39(4), 753-773.
- de Souza, C. S. (2005). *The semiotic engineering of human-computer interaction*. Cambridge, MA: The MIT Press.
- de Souza, C. S., & Leitão, C. F. (2009). *Semiotic Engineering Methods for Scientific Research in HCI*. San Rafael, CA: Morgan and Claypool Publishers.
- de Souza, C. S., Leitão, C. F., Prates, R. O., & da Silva, E. J. (2006). The semiotic inspection method. In *Proc. of the 7th Brazilian Symposium of Human Factors on Computer Systems* (pp. 148-157). Porto Alegre, Brasil: SBC.
- del Galdo, E. M., & Nielsen, J. (Eds.). (1996). *International Users Interfaces*. New York, NY: John Wiley & Sons.

- Dittrich, Y. (1997). *Computeranwendungen und sprachlicher Kontext*. Frankfurt, Germany: Peter Lang.
- Dittrich, Y. (1998). *Developing a language for participation*. Research report 18/98. University of Karlskrona/Ronneby, Sweden.
- Dittrich, Y., Vaucouleur, S., & Giff, S. (2009). ERP Customization as Software Engineering: Knowledge Sharing and Cooperation. *IEEE Softw.*, 26(6), 41-47.
- Dix, A., Finlay, J., Abowd, G., & Beale, R. (2004). *Human-Computer Interaction*. London, UK: Prentice Hall.
- Dormann, C., & Chisalita, C. (2002). Cultural Values in Web Site Design. In *Proc. of ECCE-11* (pp. 8-11).
- Dourish, P., Edwards, W. K., Lamarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D. B., & Thornton, J. (2000). Extending Document Management Systems with User-Specific Active Properties. *ACM TOIS*, 18(2), 140-170.
- Dourish, P., Anderson, K., & Nafus, D. (2007). Cultural Mobilities: Diversity and Agency in Urban Computing. In C. Baranauskas, P. Palanque, J. Abascal, & S. D. J. Barbosa (Eds.), *Proc. of INTERACT 2007* (pp. 100-113). Berlin, Germany: Springer-Verlag.
- ECMAScript (n.d.). ECMAScript Programming Language. Retrieved from <http://www.ecmascript.org/>
- Ellis C. A., Gibbs S. J., & Rein G. (1991). Groupware: some issues and experiences. *CACM*, 34(1), 39-58.
- Engelbart, D. C. (1995). Toward augmenting the human intellect and boosting our collective IQ. *Commun. ACM*, 38(8), 30-32.
- Eriksson, J. (2008). *Supporting the cooperative design process of end-user tailoring*. PhD dissertation. Blekinge Institute of Technology.
- Esselink, B. (2000) *A Practical Guide to Localization*. Amsterdam, NL: John Benjamins Publishing Company.
- EUD-Net (n.d.). EUD-Net Network of Excellence. Retrieved from <http://giove.cnuce.cnr.it/eud-net.htm>

- eXist (n.d.). eXist-db Open Source Native XML Database. Retrieved from <http://exist.sourceforge.net/>
- Fischer, G., McCall, R., Ostwald, J., Reeves, B., & Shipman, F. (1994). Seeding, evolutionary growth and reseeding: supporting the incremental development of design environments. In B. Adelson, S. Dumais, & J. Olson (Eds.), *Proc. of CHI '94* (pp. 292-298). New York, NY: ACM Press.
- Fischer, G. (2000). Social Creativity, Symmetry of Ignorance and Meta-design. *Knowledge-Based Systems Journal*, 13 (7-8), 527-37.
- Fischer, G. (2001). Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems. In S. Bjornestad, R. Moe, A. Morch, & A. Opdahl (Eds.), *Proc. of IRIS 2001* (pp. 1-14).
- Fischer, G. (2002). Beyond ‘Couch Potatoes’: From Consumers to Designers and Active Contributors”. *FirstMonday*. Retrieved from http://firstmonday.org/issues/issue7_12/fischer/
- Fischer, G. (2009). End-User Development and Meta-design: Foundations for Cultures of Participation. In V. Pipek, M. B. Rosson, B. de Ruyter, & V. Wulf (Eds.), *End-User Development* (pp. 3-14). Berlin, Germany: Springer.
- Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A., & Mehandjiev, N. (2004). Meta-design: a manifesto for end-user development. *CACM*, 47(9), 33-37.
- Fogli, D., Fresta, G., Marcante, A., & Mussio, P. (2004). IM²L: A User Interface Description Language Supporting Electronic Annotation. In *Proc. of AVI 2004* (pp. 135-142).
- Fogli, D., Marcante, A., Mussio, P., Oliveri, E., Padula, M., & Scaioli, R. (2005) Building yard on line: a distributed and mobile system for supporting building workers. In *Proc. of WETICE-2005*. IEEE Computer Society.
- Fogli, D., Mussio, P., Marcante, A., Oliveri, E., & Padula, M. (2006). Multimodal Interaction for Managing Knowledge on the Field. In *Proc. of the 1st IEEE workshop on Multimodal and Pervasive Services*.

- Fogli, D., & Piccinno, A. (2005). Environments to support context and emotion aware visual interaction. *IJVL*, 16, 386-405.
- Folmer, E., van Welie, M., & Bosch, J. (2005). Bridging Patterns: An Approach to Bridge Gaps between SE and HCI. *Journal of Information and Software Technology*, 48(2), 69-89.
- Fussell, D., & Haaland, A. (1978). Communication with pictures in Nepal: results of practical study used in visual education. *Educational Broadcasting International*, 11(1), 25-31.
- Garland, K. (1982). The use of short term feedback in the preparation of technical and instructional illustration. In *Proc. of Research in Illustration*, part II.
- Gennari, J. H., & Reddy M. (2000). Participatory Design and an Eligibility Screening Tool. In Overhage, J. M. (Ed.), *Proc. of the AMIA Annual Fall Symposium* (pp. 290-294). Philadelphia, PA: Hanley & Belfus, Inc.
- Germonprez, M., Hovorka, D., & Collopy, F. (2007). A Theory of Tailorable Technology Design. *Journal of the Association for Information Systems*, 8(6), 315-367.
- Ginige, J. A., De Silva, B., & Ginige, A. (2005). Towards End User Development of Web Applications for SMEs: A Component Based Approach. In *Proc. of the 5th International Conference on Web Engineering* (pp. 489-499).
- Ginige, A., & De Silva, B. (2009). Aspects Based Modeling of Web Applications to Support Co-evolution. In Filipe, J., Shishkov, B., Helfert, M., & Maciaszek, L. A. (Eds.) *Software and Data Technologies* (pp. 283-293). Berlin, Germany: Springer.
- Green, T. R. G., & Petre, M. (1996). Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *JVL*, 7(2), 131-174.
- Greenbaum, J., & Kyng, M. (Eds.). (1991). *Design at work: Cooperative design of computer systems*. Hillsdale, NJ: Lawrence Erlbaum.
- Grudin J. (1994). Computer-supported cooperative work: history and focus. *IEEE Computer*, 27(5), 19-26.
- Hall, E. (1959). *The silent language*. New York, NY: Doubleday.

- Harris, R. (2000). *Rethinking Writing*. London, UK: The Athlone Press.
- Hartson, H.R., & Hix, D. (1989). Human-computer interface development: concepts and systems for its management. *ACM Computing Surveys*, 21(1), 5-92.
- Hayes, J. R. (1985). *Three Problems in Teaching General Skills*. Hillsdale, NJ: Lawrence Erlbaum.
- Henderson, A., & Kyng, M. (1991). There's No Place Like Home: Continuing Design in Use. In J. Greenbaum, & M. Kyng (Eds.), *Design at Work: Cooperative Design of Computer Systems* (pp. 219-240). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Hofstede, G. (1991). *Cultures and organisations: software of the mind*. New York, NY: McGraw Hill.
- Hoft, N. L. (1996). Developing a cultural model. In E. M. del Galdo, & J. Nielsen (Eds.), *International Users interfaces* (pp. 41-73). New York, NY: Wiley & Sons.
- HTML (n.d.). XHTML2 Working Group Home Page. Retrieved from <http://www.w3.org/MarkUp/>
- HTTP (n.d.). HTTP – Hypertext Transfer Protocol. Retrieved from <http://www.w3.org/Protocols/>
- Hutchins, E. L., Hollan, J. D., & Norman, D. (1986). Direct manipulation interfaces. In D. Norman, & S. Draper (Eds.), *User Centred System Design* (pp. 87-124). Hillsdale, NJ: Lawrence Erlbaum.
- Iacob, C. (2010). *Design Patterns and Interactive Systems for Supporting Creative Collaborative Design Processes*. Ongoing PhD Project. Università degli Studi di Milano.
- ISO 639-1 (n.d.). Codes for the representation of names of languages - Part 1. Retrieved from <http://www.iso.org/>
- ISO 3166-1 (n.d.). Codes for the representation of names of countries and their subdivisions - Part 1. Retrieved from <http://www.iso.org/>

- Jennings, P. (2005). Tangible social interfaces: critical theory, boundary objects and interdisciplinary design methods. In *Proc. of C&C '05* (pp. 176-186). New York, NY: ACM Press.
- Johansen, R. (1988). *Groupware: Computer Support for Business Teams*. New York, NY: The Free Press.
- Kirsh, D. (1995). The Intelligent Use of Space. *Artificial Intelligence*, 73(1-2), 31-68.
- Ko, A., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M. B., Rothermel, G., Shaw, M., & Wiedenbeck (2010). The State of the Art in End-User Software Engineering. *Accepted for publication on ACM Computing Surveys*.
- Koivunen, M. R., & Swick, R.R. (2003). Collaboration through annotations in the Semantic Web. In S. Handschuh, & S. Staab (Eds.), *Annotation for the Semantic Web* (pp. 46-60). Amsterdam, The Netherlands: IOS Press.
- Konkola, R. (2001). Developmental process of internship at polytechnic and boundary-zone activity as a new model for activity (in Finnish). In T. Tuomi-Gröhn, Y. Engeström, & M. Young (Eds.), *At the bounday zone between school and work – new possibilities of work-based learning* (pp. 148-186). Helsinki, Finland: University Press.
- Kuutti, K. (1996). Activity Theory as a Potential Framework for Human-Computer Interaction Research. In B. Nardi (Ed.), *Context and Consciousness: Activity theory and human-computer interaction* (pp. 17-44). Cambridge, MA: MIT Press.
- Kyng, M. (1991). Designing for cooperation: cooperating in design. *CACM*, 34(12), 65-73.
- Lauesen, S. (2005). *User Interface design – A Software Engineering Perspective*. Reading, MA: Addison-Wesley.
- Lehman, J. A. (1985). Personal computing vs. personal computers. *Information and Management*, 9(5), 253-259.
- Lehman, M. M., Ramil, J. F., Wernick, P. D., Perry, D. E., & Turski, W. M. (1997). Metrics and laws of software evolution—the nineties view. In *Proc. of METRICS'97* (pp. 20-32).

- Leitheiser, R., & Wetherbe, J. (1986). Service support levels: An organized approach to end-user computing. *MIS Quarterly*, 10(4), 337-349.
- Leitão, C. F., de Souza, C. S., & Barbosa, C. M. A. (2007). Face-to-face sociability signs made explicit in CMC. In *Proc. of Interact 2007* (pp. 5-18). Berlin, Germany: Springer.
- Lewis, J. R. (1993). *IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use*. Technical report 54.786. IBM.
- Lieberman, H., Paternò, F., Klann, M., & Wulf, V. (2006). End-User Development: An Emerging Paradigm. In H. Lieberman, F. Paternò, & V. Wulf (Eds.), *End-User Development* (pp. 1-8). Dordrecht, The Netherlands: Springer.
- LISA (2010). Retrieved from <http://www.lisa.org>
- Madell, T., Parson, C., & Abegg, J. (1994). *Developing and localizing international software*. Upper Saddle River, NJ: Prentice-Hall, Inc.
- Mahemoff, M. J., & Johnston, L. J. (1998). Software Internationalisation: Implications for Requirements Engineering. In D. Fowler, & L. Dawson (Eds.), *Proc. of the Third Australian Workshop on Requirements Engineering* (pp. 83-90). : Geelong, Australia: Deakin University.
- Majhew, D. J. (1992). *Principles and Guideline in Software User Interface Design*. Englewood Cliffs, NJ: Prentice Hall.
- Maldonado, H., Lee, B., Klemmer, S. R., & Pea, R. D. (2007). Patterns of Collaboration. In *Design Courses: Team dynamics affect technology appropriation, artifact creation, and course performance* (pp. 490-499). ISLS.
- Marcante, A., & Mussio, P. (2006). Electronic Interactive Documents and Knowledge Enhancing: a Semiotic Approach. *Presented at the Document Academy*. Berkeley, CA: UC Berkeley.
- Marcante, A., & Parasiliti Provenza, L. (2008). Social Interaction through Map-based Wikis. *PsychNology Journal*, 6(3), 247-267.

- Marcus, A. (2001). Cultural Dimensions and Global Web Design: What? So What? Now What? In *Proc. of Human Factors and the Web* (pp. 1-15).
- Merriam-Webster online (2010). Retrieved October 20th, 2010, from <http://www.merriam-webster.com>.
- Miller, N. G. (2000). *Task management system*. U.S. Patent 6,101,481.
- Minazzi, F. (2008). *Lecture on software internationalization and localization*. Department of Information and Communication, Università degli Studi di Milano, Italy. Retrieved from <http://webcen.dsi.unimi.it/wcinfo/attivi060708.html>
- Mitamura, Y. K. (1985). *Let's learn hiragana*. Tokyo, Japan: Kodansha International.
- Mitamura, J. Y., & Mitamura, Y. K. (1997). *Let's learn kanji*. Tokyo, Japan: Kodansha International.
- Mørch, A. (1997). Three levels of end-user tailoring: customization, integration, and extension. In M. Kyng, & L. Mathiassen (Eds.), *Computers and Design in Context* (pp. 51-76). Cambridge, MA: The MIT Press.
- Murty, K. S. (2010). Building and Sustaining Communities of Practice. AOK Association of Knowledgework. Retrieved October 20th, 2010, from http://kwork.org/white_papers/
- Mussio, P. (2009). *Lectures on Human-Computer Interaction*. Università degli Studi di Milano.
- Mussio, P., Pietrogrande, M., & Protti, M. (1991). Simulation of hepatological models: a study in visual interactive exploration of scientific problems. *IJVL*, 2(1), 75-95.
- Mussio, P. (2003). E-Documents as tools for the humanized management of community knowledg. In H. Linger, et al. (Eds.), *Constructing the Infrastructure for the Knowledge Economy: Methods and Tools; Theory and Practice* (pp. 27-42). New York, NY: Kluwer.
- Nake, F., & Grabowski, S. (2001). Human-computer interaction viewed as pseudo-communication. *Knowledge-Based System*, 14, 441-447.
- Nardi, B. (1993). *A Small Matter of Programming*. Cambridge, MA: The MIT Press.

- Nardi, B. A., & Miller, J. R. (1990). An ethnographic study of distributed problem solving in spreadsheet development. In *Proc. of CSCW '90* (pp. 197-208). New York, NY: ACM Press.
- Nielsen, J. (1993). *Usability Engineering*. San Diego, CA: Academic Press.
- Nielsen, J. (1994). Heuristic evaluation. In J. Nielsen, & R. L. Mack (Eds.), *Usability Inspection Methods* (pp. 25-64). New York, NY: John Wiley & Sons.
- Nielsen, J. (1996). *International Web Usability*. useit.com alertbox, August. Retrieved from <http://www.useit.com/alertbox/9608.html>
- Nunamaker, J. F., Chen, M., & Purdin, T. D. M. (1991). System Development in Information Systems Research. *Journal of Management Information Systems*, 7(3), 89-106.
- O'Hagan, M., & Ashworth, D. (2002). *Translation-mediated communication in a digital world Facing the challenges of globalization and localization*. Clevedon, UK: Multilingual Matters LTD.
- O'Reilly, T. (2006). Web 2.0 Compact Definition: Trying Again. Retrieved from <http://radar.oreilly.com/2006/12/web-20-compact-definition-tryi.html>
- Ogden C. K., & Richards I. A. (1960). *The Meaning of Meaning. A Study of the Influence of Language upon Thought and of the Science of Symbolism*. London, UK: Routledge & Kegan Paul.
- OWL (n.d.). Web Ontology Language Overview W3C Recommendation. Retrieved from <http://www.w3.org/TR/owl-features>
- Penner, R. R., & Steinmetz, E. S. (2002). Model-Based Automation of the Design of User Interfaces to Digital Control Systems. *IEEE TSMCA*, 32(1), 41-49.
- Perkowitz, M., & Etzioni, O. (2000). Towards adaptive Web sites: Conceptual framework and case study. *Artificial Intelligence*, 118(2000), 245-275.
- Petre, M. (1995). Why looking isn't always seeing: Readership skills and graphical programming. *CACM*, 38(6), 33-44.

- Petre, M., & Blackwell, A. F. (2007). Children as Unwitting End-User Programmers. In *Proc. of VL/HCC 2007* (pp. 239-242).
- PHP (n.d.). PHP: Hypertext Preprocessor. Retrieved from <http://php.net/>
- Piccinno, A. (2004). *Software environments for supporting End-User Development*. PhD thesis. Università degli Studi di Bari, Italy.
- Pipek, V., Rosson, M.B., de Ruyter, B., & Wulf, V. (2009). In Pipek, V., Rosson, M.B., de Ruyter, B., & Wulf, V. (Eds.), *Proc. of IS-EUD 2009* (pp. V-VI). Heidelberg, Germany: Springer.
- Polanyi, M. (1967). *The tacit dimension*. London, UK: Roulledge & Kegan Paul.
- Prates, R. O., de Souza, C. S., & Barbosa, S. D. J. (2000). A method for evaluating the communicability of user interfaces. *ACM Interactions*, 7(1), 31-38.
- Prates, R. O., Barbosa, S. D. J., & de Souza, C. S. (2000). A case study for evaluating interface design through communicability. *Proc. of DIS2000* (pp. 308-317). New York, NY: ACM Press.
- RDF (n.d.). Resource Description Framework. Retrieved from <http://www.w3.org/RDF/>
- Reinecke, K., & Bernstein, A. (2008). Predicting user interface preferences of culturally ambiguous users. In M. Burnett, M. F. Costabile, T. Catarci, B. de Ruyter, D. Tan, M. Czerwinski, & A. Lund (Eds.), *Proc. of CHI '08* (3261-3266). New York, NY: ACM Press.
- Repenning, A. (1991). Creating User Interfaces with Agentsheets. In *Proc. of 1991 Symposium on Applied Computing* (pp. 190-196). Los Alamitos, CA: IEEE Computer Society Press.
- Repenning, A., & Ioannidou, A. (2006). What makes End-user development tick? 13 Design Guidelines. In H. Lieberman, F. Paternò, & V. Wulf (Eds.), *End-User Development* (pp. 51-85). Dordrecht, The Netherlands: Springer.
- Resnick, L. B., Levine, J. M., & Teasley, S. D. (1991). *Perspectives on socially shared cognition*. Washington, DC: American Psychology Association.

- Rittel, H. (1984). Second-Generation Design Methods. In N. Cross (Ed.), *Developments in Design Methodology* (pp. 317-327). Chichester, UK: John Wiley & Sons, Ltd.
- Robinson, M. (1993). Design for unanticipated use. In G. De Michelis, C. Simone, & K. Schmidt (Eds.), *Proc. of ECSCW '93* (pp. 187-202). Dordrecht, The Netherlands: Springer.
- Rockart, J. F., & Flannery, L. S. (1983). The Management of End-User Computing. *CACM*, 26(10), 776-784.
- Russo, P., & Boor, S. (1993). How fluent is your interface?: designing for international users. In S. Ashlund, A. Henderson, E. Hollnagel, K. Mullet, & T. White (Eds.), *Proc. of INTERCHI '93* (pp. 342-347). Amsterdam, NL: IOS Press.
- Savourel, Y. (2001). *XML Internationalization and Localization*. Indianapolis, IN: SAMS.
- Scala, P. (2010). *Progettazione e sviluppo di un prototipo di un Task Management System per la gestione multimodale dell'informazione*. M.Sc. thesis. Advisor: Piero Mussio. Co-Advisors: Barbara Rita Barricelli, Marco Padula.
- Schön, D. (1983). *The Reflective Practitioner: How Professionals Think in Action*. London, UK: Maurice Temple Smith.
- Schuler, D. (2008). *Liberating Voices: a Pattern Language for Communication Revolution*. Cambridge, MA: The MIT Press.
- Schuler, D., & Namioka, A. (Eds.). (1993). *Participatory Design: Principles and Practices*. Hillsdale, NJ: L. Erlbaum Associates Inc.
- Sharp, H., Rogers, Y., & Preece, J. (2007). *Interaction design: Beyond human computer interaction*. Hoboken, NJ: Wiley.
- Shneiderman, B. (2002). *Leonardo's Laptop: Human Needs and the New Computing Technologies*. Cambridge, MA: The MIT Press.
- Smith, A., & Chang, Y. (2003). Quantifying Hofstede and Developing Cultural Fingerprints for Website Acceptability. In V. Evers, K. Rose, P. Honold, J. Coronado, & D. L. Day (Eds.), *Proc. of IWIPS 2003* (pp. 89-102). London, UK: P&SI.

- Snow, C.P. (1959). *The Two Cultures and the Scientific Revolution*. Cambridge, MA: Cambridge University Press.
- Software Engineering Institute (2010): Ultra-Large-Scale Systems: The Software Challenge of the Future. Retrieved October 20th, 2010, from <http://www.sei.cmu.edu/uls/>
- Spahn M., Doerner C., & Wulf, V. (2008). End User Development: Approaches Towards a Flexible Software Design. In W. Golden, T. Acton, K. Conboy, H. van der Heijden, & V. K. Tuunainen (Eds.), *Proc. of ECIS 2008* (pp. 303-314).
- Stamper, R., Liu, K., Hafkamp, M., & Ades, Y. (2000). Understanding the Roles of Signs and Norms in Organisations: A semiotic approach to information system design. *Journal of Behaviour & Information Technology*, 19(1), 12-27.
- Star, S. L., & Griesemer, J. R. (1989). Translations' and Boundary Objects: Amateurs and Professionals in Berkley's Museum of Vertebrate Zoology 1907-39. *Social Studies of Science*, 19(3), 387-420.
- Stry, C. (2000). TADEUS: Seamless Development of Task-Based and User-Oriented Interfaces. *IEEE TSMCA*, 30(5), 509-525.
- Stevens, G., Quaisser, G., & Klann, M. (2006). Breaking It Up: An Industrial Case Study of Component-Based Tailorable Software Design. In H. Lieberman, F. Paternò, & V. Wulf (Eds.), *End-User Development* (pp. 269-294). Dodrecht, Germany: Springer.
- Sutcliffe, A., & Mehandjiev, N. (2004). Introduction of Special Issue on End-User Development. *CACM*, 47(9), 31-32.
- SVG (n.d.). Scalable Vector Graphics. Retrieved from <http://www.w3.org/Graphics/SVG/>
- Thorell, L. G., & Smith, W. J. (1990). *Using Computer Color Effectively: An Illustrated Reference*. Englewood Cliffs, NJ: Prentice Hall.
- Tondl, L. (1981). *Problems of Semantics*. Boston, MA: Reidel Publishing.
- Trigg, R.H., Moran, T.P., & Halasz, F.G. (1987). Adaptability and Tailorability in NoteCards. In B. Shackel, & H-j. Bullinger (Eds.), *Proc. of INTERACT'87* (pp. 723-728). Amsterdam, The Netherlands: Elsevier Science Publishers.

- Trompenaars, F. (1993). *Riding the waves of culture*. London, UK: Nicholas Brealey publishing.
- Vaishnavi, V., & Kuechler, W. (2004). *Design Research in Information Systems*. Retrieved from <http://home.aisnet.org/displaycommon.cfm?an=1&subarticlenbr=794>
- Valtolina, S. (2008). Design of Knowledge Driven Interfaces in Cultural Contexts. *IJSC*, 2(4), 525-553.
- Valtolina, S. (2010). *Lectures on Fundamentals of Digital Communication*. Università degli Studi di Milano.
- Verplank, B. (2009, September). *Interaction Design Sketchbook*. Lecture held at the First International DESIRE Summer School Theories of Creative Design for Innovation in Science and Technology. Università degli Studi di Milano, Gargnano del Garda, Italy.
- Victor, D. (1992). *International business communications*. New York, NY: Harper Collins.
- Wagner E. L., & Piccoli G. (2007). Moving beyond user participation to achieve successful IS design. *CACM*, 50(12), 51-55.
- Wenger, E. (1998). *Communities of Practice. Learning, Meaning, and Identity*. Cambridge, MA: Cambridge University Press.
- Whitley, K. N., & Blackwell, A. F. (2001). Visual Programming in the Wild: A Survey of LabVIEW Programmers. *JVLC*, 12(4), 435-472.
- Wright, M., Marlino, M., & Sumner, T. (2002). Meta-design of a community digital library. *D-Lib Magazine* 8(5). Retrieved from <http://www.dlib.org>.
- WSDL (n.d.). Web Service Description Language Version 2.0 W3C Recommendation. Retrieved from <http://www.w3.org/TR/wsdl20>
- XPath (n.d.). XML Path Language. Retrieved from <http://www.w3.org/TR/xpath/>
- XPointer (n.d.). XML Pointer Language. Retrieved from <http://www.w3.org/TR/WD-xptr>
- Ye, Y., & Fischer, G. (2007). Designing for Participation in Socio-Technical Software Systems. In C. Stephanidis (Ed.), *Proc. of UAHCI* (pp. 312-321). Heidelberg, Germany: Springer.

- Yeo, A. (1996). Cultural user interfaces: a silver lining in cultural diversity. *SIGCHI Bull.*, 28(3), 4-7.
- Yeo, A. W. (2001). Global-software development lifecycle: an exploratory study. In *Proc. of CHI '01* (pp. 104-111). New York, NY: ACM Press.
- Zhu, L. (2010). *A meta-design framework to support creative collaborative software-design by distributed multidisciplinary teams*. Ongoing PhD Project. Università degli Studi di Milano.
- Zhu, L., Mussio, P., & Barricelli, B. R. (2010). Hive-Mind Space Model for Creative Collaborative Design. In *Proc. of DESIRE'10* (pp. 121-130). New York, NY: ACM Press.
- Zhu, L., Mussio, P., Barricelli, B. R., & Iacob, C. (2010). A Habitable Space for Supporting Creative Collaboration. In *Proc. of CTS2010* (pp. 617-622). Los Alamitos, CA: IEEE Computer Society.